Database Support for Uncertain Data Analysis and Correlation Handling in Scenario Planning

vorgelegt von
Dipl.-Inf.
Katrin Eisenreich
geboren am 29.12.1983 in Rodewisch

von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Tag der wissenschaftlichen Aussprache: 22. Oktober 2012

Berlin 2013
D 83

Statutory Declaration

I declare that I have authored this thesis entirely independently, that I have not used any other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. This thesis has never been used for graduation at any academic institution.

Dresden, July 4th 2012

Date

Signature

Abstract

Today, the environment in which companies devise their plans for the future evolves at a fast pace. In a planning process, one must keep in mind the highly volatile, and therefore uncertain factors that determine whether a goal can be achieved and what are the associated risks. One approach to address this issue is scenario planning—a methodology that allows experts to evaluate different possible scenarios of the future based on assumptions about relevant business influences.

Existing tools for planning and risk analysis partially support scenario planning. However, their application-specific implementation of data models and operators can come at the cost of inefficient processing and hinder the integration between applications. First, since all processing steps are carried out on the application level, especially data-intensive operations incur high additional costs for data transfer at runtime. Second, since the resulting scenario data is stored and accessed in an application-specific manner, it is not easily accessible from other applications. An approach to address those problems is to provide database support for representing and processing data in a scenario planning process. Applications can access this functionality as building blocks to assemble specific analysis and planning processes, while individual calculation steps are carried out at the database.

This thesis addresses especially the aspect of handling *uncertainty* in scenario planning processes, with focus on the analysis of continuously distributed data at the database level. At the core of the thesis, a probabilistic data model is used to address the representation of uncertain scenario data. Building on this model, a set of analytical operators for the creation and analysis of such data is provided.

In this context, we specifically focus on the flexible analyses of *correlation structures* in data. Correlation is a highly relevant aspect when analyzing risks or chances associated with business activities, since most real-world developments are subject to complex dependencies. Therefore, it is crucial to enable users to flexibly evaluate *arbitrary* correlation structures and their impact on potential outcomes of a plan. To this end, an approach for modeling correlation structures at the database and efficient methods for their construction and processing are presented in this thesis. The proposed techniques go beyond existing approaches for correlation handling in probabilistic databases, supporting the analysis of complex correlations between arbitrary distributions in a modular and flexible manner.

To enable both the interpretation and the efficient *recomputation* of derived scenario data, the thesis further addresses the topic of provenance for scenario data. In particular, an approach for exploiting provenance information for a selective, and thus more efficient, recomputation of analysis results is presented and evaluated.

In summary, this thesis addresses both data- and process-centric aspects of uncertainty in scenario planning processes. The presented approaches thus go beyond the scope of many existing solutions in the field of probabilistic data management, which mostly focus on managing and processing uncertain data in a traditional relational setting. The developed functionalities can serve applications in the field of strategic and operational planning, predictive analytics, and enterprise and risk management as a foundation for an efficient and flexible processing of uncertain data in scenario planning processes.

Contents

Acronyms

ACR Approximate Correlation Representation. 64–78, 96, 97, 99, 100, 108–111, 115, 119, 120, 126, 129–132, 138–144, 151, 154, 158, 161–163

BI Business Intelligence. 13

CDF cumulative distribution function. 25, 26, 31, 32, 43, 46, 93, 115, 116, 118

EPM Enterprise Performance Management. 13

GSL GNU Scientific Library. 151, 154

ICDF inverse cumulative distribution function. 26, 31, 43, 116

LDB Lineage Database. 107

MC Monte Carlo. 24, 25, 95–97, 111–115, 118–121, 123, 151, 152, 154, 155, 157–159, 162

MCDB Monte Carlo Data Base. 106, 112–115, 119–121, 123, 151–157

MCMC Markov Chain Monte Carlo. 97, 109

OLAP Online Analytical Processing. 8, 39, 44, 67, 124, 125

OPM Open Provenance Model. 37, 83, 84, 105, 106

PDB probabilistic database. 21, 31, 32, 34, 98, 99, 158, 162

PDF probability density function. 25, 26, 30–32, 43, 46, 115, 118

PGM Probabilistic Graphical Model. 98–100, 108–110, 118, 119, 163

PMF probability mass function. 25

PW possible world. 21, 30, 159

PWS Possible World Semantics. 21, 23, 26, 34, 36

ULDB Uncertainty and Lineage Database. 106, 107

VE Variable Elimination. 99, 109, 119

VG Variable Generation. 112–115, 117, 119, 121, 123, 151–154, 159

List of Figures

List of Tables

1 Introduction

The world today evolves at a faster pace, and so does the environment in which a company, or any organization which strives for performance, devises its plans for the future. In a planning process, it is therefore necessary to keep in mind the more volatile and therefore less predictable economic, ecologic, and social factors that determine how a strategy or a plan will meet the requirements of the future. In other words, there are many possible *scenarios* that should be taken into account when devising plans for sales activities, development projects, or other business endeavors. One must identify and analyze the uncertainty, and therefore, the risk that is associated with assumptions and related scenarios of how the business may develop. This thesis focuses on the aspect of *uncertain data* in scenario planning processes, and addresses its flexible representation and processing at database level.

The method of *scenario planning* was added to the management portfolio in the 1970s, extending the traditional methods of business forecasting. Rather than predicting a single probable state of the future business, scenario planning considers different uncertain aspects to derive a number of possible, even though less probable, future states. This way, as [Wac85] puts it, scenario planning helps to "accept uncertainty, try to understand it, and make it part of our reasoning."

Apart from tools and libraries for generic data analysis tasks, there exist a range of tools that support users in prediction making and scenario planning. This includes spreadsheet-based solutions for scenario creation, but also custom applications that provide functionality for strategic, financial, and operational planning and risk management. Such solutions allow users, for example, to specify drivers and their possible valuations as input to a model of a specific business aspect and evaluate the corresponding outcomes of the model. The computations that are involved in such a model can be complex, including the application of different drivers and causalities between those drivers. Further, the provided operations potentially require access to a variety of data, such as historic data which serves as input for a model or results of economic forecasts.

Ideally, the management of risk and performance in a company should be implemented in an integrated framework [Bos06]. In order to allow different applications to access, process, and make available various kinds of data in a scenario analysis process, they should operate on a common data schema. Furthermore, the semantics and processing of basic operations applied across different applications should be the same, and their implementation should be efficient. However, in reality, different applications for data analysis, scenario planning, report creation, etc. are often not integrated. They operate on data silos (e.g., different databases or spreadsheets), applying statistical analysis functionality implemented at application level. This introduces problems regarding both the consistency of data and the semantics of provided operators, as well as the efficiency of those operators. To circumvent those problems, this thesis addresses the implementation of scenario planning functionalities at the database level, in order to enable applications to access, create, and adapt scenario data efficiently. In this regard, the thesis focuses on the handling of *uncertainty* in data as a central aspect of scenario planning. It considers both the representation and the processing of uncertain data, covering the modeling of different aspects of uncertainty and a basic set of analytic functionality for processing the uncertain information.

The uncertainty in planning processes involves many facets that need to be addressed by a model for uncertain data. Those include the imprecision of estimations and forecasting results, temporally uncertain developments of processes, and potentially complex dependencies that exist in most real-world data. There already exist a broad range of approaches to uncertain data management, i.e., approaches addressing the representation, storage, and processing of uncertain data. Most of those solutions are, however, situated in application fields such as information extraction, sensor data management, or artificial intelligence. They therefore mostly focus on the representation and querying of (or inference over) existing uncertain data. Examples include uncertain information extraction results and imprecise sensor measurements. Besides analyzing uncertain data, the scenario planning context makes it necessary to also enable users to *introduce and modify uncertain data* that may follow arbitrary distributions, as well as to manage and *evolve* the derived scenario data. Those points, which have been addressed only to a lesser degree by existing work, are covered in this thesis. As central topics, this includes the analysis of multivariate distributions which may be created by users based on assumptions about underlying dependencies, and the recomputation of analysis results under changed input data.

After having given a first insight into the core topic of this thesis, the remainder of this chapter introduces the reader more closely to the context of scenario planning and the requirements for handling the uncertainty in this context.

1.1 Scenario Planning

To devise plans for a business development, one must take decisions about activities that shall be taken in order to achieve set goals. Examples for such activities are, e.g., the introduction of a new product in a product portfolio, or investments in a specific project, development effort, or marketing activity. The information considered in a planning process can be manifold, including available historic data (e.g., previous sales numbers), data reflecting forecasts of an economic figure (e.g., concerning the economic growth in a region), or experts' implicit knowledge and expectations. In any case, the goal is to take actions in such a way as to achieve an optimal outcome with respect to the set goal. For example, when planning a product portfolio, the goal could be to maximize the total revenue gained by all products in the portfolio throughout the year.

In a traditional approach to planning and forecasting, future developments are forecast based on a potentially complex algorithm, and the outcome is then used as a basis for decision-making. However, this method does not account for changes and possible rapid shifts in the business environment. Considering only one probable future state of the world becomes inappropriate once we are looking further into the future and the potential materializations of the influential aspects change. Therefore, the planner must take into consideration that also the less expected possibilities can indeed materialize. The *uncertainty* that comes along with different possible developments must be addressed and taken into consideration in the planning process. It must be explicitly introduced in the analysis and managed appropriately in order to facilitate its interpretation by the user. Even the most advanced forecasting algorithm or a highly-experienced analyst cannot predict the future with absolute certainty. Considering a forecast as a crisp fact clearly can mislead its interpretation, as well as any further results derived from the forecast data during an analysis or planning process.

Rather than disguising the uncertainty, we must therefore appropriately represent it and propagate it through subsequent calculations which use the uncertain information as input. Assume an analyst wants to predict the sales of a newly developed product, for which no reliable historic data—and therefore no appropriate basis for forecasts—is available. He may consider a (supposedly) similar product as a reference, and use its sales figures as a basis to forecast the sales of the new product. In this case, the data would reflect the underlying *assumption* that the sales of the new product will behave similar to the sales of the reference product. Since this assumption cannot be verified at the time when it is made, it introduces a factor of uncertainty into the overall calculation. Another aspect of uncertainty is that, in the real-world, developments are often a result of different factors that *depend* on each other. For example, consider the economic growth and the disposable budget of consumers. Usually, those factors will not develop independently from each other. Rather, an increase in the economic growth rate will likely lead to rising incomes, which in turn can increase the amount of money consumers spend. An analyst must take into consideration the influence of such dependencies since they can have a great influence on the outcomes of a business development.

Being able to predict a range of possible outcomes enables the user to evaluate the chance that a goal will be achieved, or conversely, the risk that a goal will not be achieved. Then, he can devise measures to increase chances or decrease risks. In this way, scenario planning reinforces the insight into opportunities and risks in business developments. Consequently, it can help to manage both aspects in a more informed way. The benefit of using scenarios has been acknowledged in the industry, and major vendors meanwhile provide specialized products for planning and risk analysis in different domains. As stated above, approaches for managing uncertainty at database level do mostly not address the scenario analysis context. Those solutions that *do* incorporate the concept of 'what-if'-analysis yet lack important features, such as the flexible handling of dependencies in data.

This section described the application context within which the thesis is situated. In the following section, we now turn to two basic use cases that illustrate the requirements of uncertainty management in scenario planning.

1.2 Example Use Cases

The following use cases shall serve the reader to understand specific aspects of uncertainty in scenario planning, show where uncertainty exists in the underlying data, and how it may be introduced or increased. At this point, the use case descriptions are at a conceptual level and specific characteristics of distributions that underly the data are not considered. The descriptions will be refined during Chapter 2, where the use cases serve to exemplify various aspects that build the foundation of this thesis.

1.2.1 Use Case 1: Sales Planning

The first use case considers the planning of future sales amounts for a given product. In this use case, uncertainty plays a role both with respect to the amount of sales that can be achieved and the time frame within which the product will be sold.

Task A: Predicting Sales Suppose a producer of consumer electronics wants to analyze the potential sales of a new smart phone $P2$, which shall be introduced to their product portfolio at the beginning of the next year (assumed to be 2012). To project prospective sales for the following year under different assumptions, a planner may consider manifold data. Both historic and forecast information can play an important role. Naturally no historic sales data is available for $P2$. Therefore, the planner wants to take the historic sales data of a similar product as reference and assume that $P2$ will generate similar sales. Additionally, he wants to consider further aspects that he assumes will have an influence on the sales of $P2$, such as the effect of the general economic development, or the influence of marketing activities.

Figure 1.1 illustrates, on a conceptual level, the assumptions the planner applies for the described task.

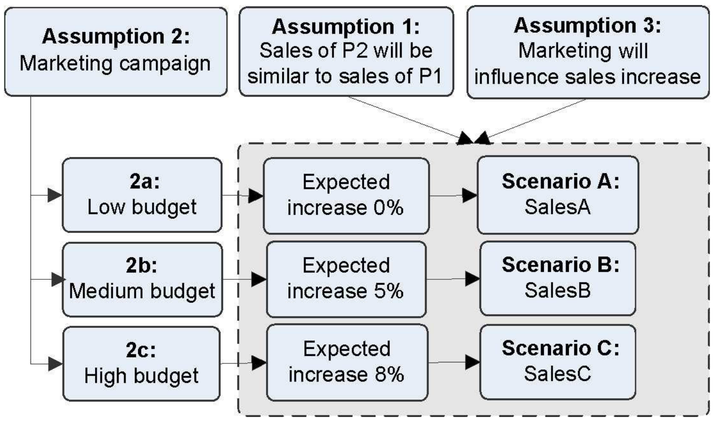


Figure 1.1: Deriving scenarios under alternative economic assumptions

First, he includes data about the current year's (2011) sales of the reference product, *P1*, in his analysis. He computes the monthly sales numbers of *P1* per store from historic data and assumes that their *distribution* will reflect the sales of *P2* in 2012. In doing so, the expert implicitly makes the assumption (Assumption 1) that the newly introduced product will behave similarly to the established product, e.g., because *P2* is a successor of *P1*. Second, the assumption is that a marketing campaign will have an impact on the sales of *P2*. The user therefore incorporates the planned marketing investment in 2012 in his analysis (Assumption 2). Finally, he introduces the assumption that the sales increase will *depend*, to some degree, on the marketing efforts (Assumption 3). To get an understanding of possible variations in future sales, the analyst then creates three scenarios reflecting alternative marketing budgets. Particularly, he considers the possibility of a worst case, i.e., a *low budget*, a likely case of a *medium budget*, and a best case of a *high budget* (*1a*, *1b*, and *1c*). Based on those three cases, three expected values for the sales increase are computed. Each of the three final scenarios A, B, and C then represents *one* possible value of the monthly sales that may be generated by *P2* under the applied assumptions.

Task B: Analysis of Temporal Developments In a second task, the analyst needs to report prospected sales figures of *P2* for the first two quarters of 2012, based on different possibilities regarding the release of *P2*. The assumptions which are introduced in his analysis are schematically depicted in Figure 1.2.

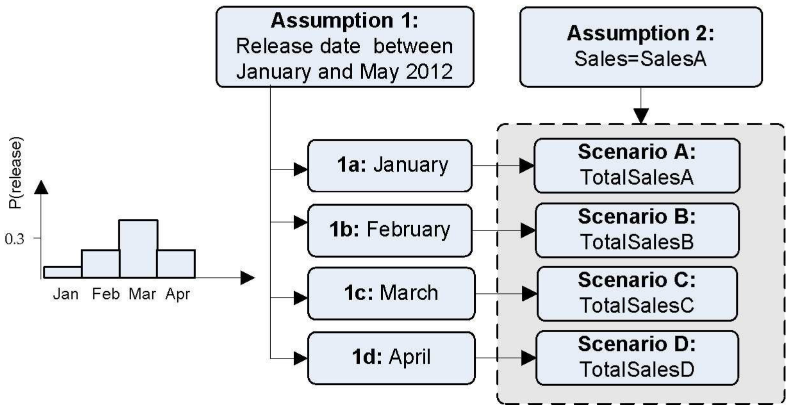


Figure 1.2: Deriving scenarios for alternative temporal developments

Suppose that, due to a very tight production plan, it is not clear when exactly *P2* will be launched. The analyst wants to understand how sales may look at the end of Q2 in 2012. To this end, he considers the launch date of *P2* to be uncertain, at some time between the beginning of January and the beginning of May (Assumption 1). For example, the launch may happen very likely at the beginning of March, and less likely much earlier or later than that, as indicated in the left-hand side of the figure. Also, the analyst assumes that the monthly sales value derived for Scenario A of the previous task will be achieved (Assumption 2). The analyst now can distinguish further scenarios for different potential release dates, e.g., at the beginning of January, February, March, or April (*1a* through *1d*), respectively, in order to compute the total sales amount potentially achieved until the end of Q2, resulting in scenarios *A – D*.

In the presented use case, the required functionalities for planning under uncertainty include the introduction of uncertainty from deterministic fact data, the computation of expectations, and the aggregation of key figures associated with events in time for which no deterministic occurrence time can be given.

1.2.2 Use Case 2: Insurance Risk Evaluation

The second use case focuses on the evaluation of risks associated with business developments. In the insurance domain, it is particularly important for an insurer to understand the occurrence of damage events that are covered by a provided insurance plan. Specifically, the insurer must have a good understanding of the risk that events cause extreme damages (i.e., very high financial loss) that may, for example, exceed the insurer's premium reserve.

Task A: Evaluating Risks for Individual Assets As a first task, suppose the analyst wants to prospect and analyze damages incurred by an insurance plan, say, against flood damages. To this end, he needs to know (or rather make assumptions about) the prospective claim amounts caused by flood-related damages. Figure 1.3 illustrates the assumptions made for this task.

Given sufficient data of past damages, the analyst can estimate or forecast expected flood damages. In the example, the assumed damage is estimated to be about $\$1b$ per month in the considered region (Assumption 1). The user can then analyze the forecast under different assumptions about the climatic development, e.g., on the basis of a climatic forecast (Assumption 2). Assuming that there is a dependence between flood damages and the climate (Assumption 3), the analyst decides to investigate the potential damages for three scenarios with low, average, or high rainfall (*2a – 2c*) in the region of interest. Finally, he can calculate the risk that an extremely high damage value, e.g., of $\$15b$ per month, will be exceeded for each of the three rainfall scenarios (Assumption 4). In this case, the analysis may result in a *very low*, *low*, and *medium* risk level reflected by scenarios *A – C*.

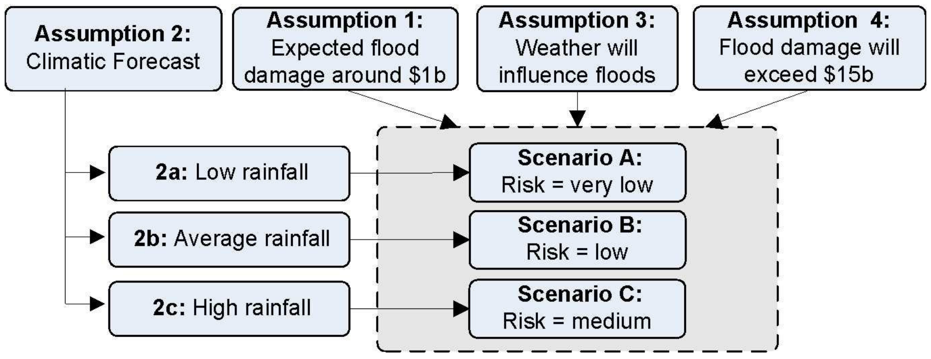

Assumption 2: Climatic Forecast
Assumption 1: Expected flood damage around $1b
Assumption 3: Weather will influence floods
Assumption 4: Flood damage will exceed $15b
2a: Low rainfall
2b: Average rainfall
2c: High rainfall
Scenario A: Risk = very low
Scenario B: Risk = low
Scenario C: Risk = medium

Figure 1.3: Deriving risk scenarios depending on weather conditions

Task B: Evaluating Joint Risks Understanding risk under dependencies becomes even more important as an insurer usually provides insurance against *different* kinds of damage, which are often correlated. Therefore, a company must evaluate how likely there will be extreme events of two or more kinds of damage: A joint occurrence of such extreme events will cause high combined claims to the insurer. In the worst case, the existence of the insurance company can be at stake as claimed benefits may easily exceed the available premium reserve. Assume that the user now wants to analyze how likely it is that there will be very high claims for two types of insurance, say against flood and hurricane damages, simultaneously. The different assumptions involved in this task are illustrated in Figure 1.4.

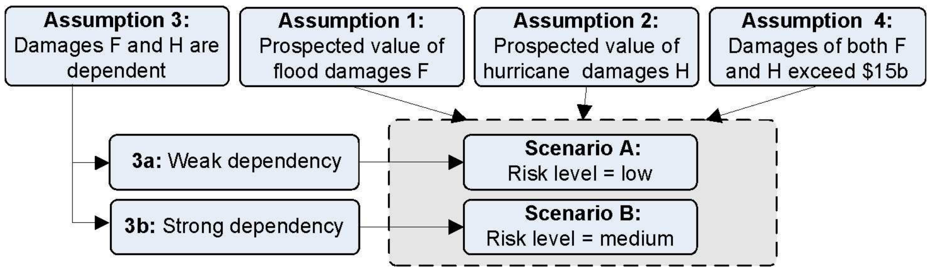

Assumption 3: Damages F and H are dependent
Assumption 1: Prospected value of flood damages F
Assumption 2: Prospected value of hurricane damages H
Assumption 4: Damages of both F and H exceed $15b
3a: Weak dependency
3b: Strong dependency
Scenario A: Risk level = low
Scenario B: Risk level = medium

Figure 1.4: Deriving risk scenarios under different correlation assumptions

First, the user prospects the expected flood and hurricane damages (Assumptions 1 & 2). Further, he assumes there is a dependency between both classes of insurance (Assumption 3). He could, for example, analyze the effect of two assumed dependency structures, where one reflects a weak dependency, and one a strong dependency (*3a, 3b*). Finally, he can evaluate the risk that both classes of insurance will cause high claim amounts over $\$15b$ each (Assumption 4). In the example, the resulting risk is *low* when applying the weak dependency assumption, and *medium* for the scenario reflecting the strong dependency.

Given the insight to those risk levels, an insurer might adjust the premium reserve in order to be able to cover worst-case losses, or decide on an appropriate counter-insurance given the computed risk is too high. This example shows that enabling the analysis of business aspects under varying dependencies can help identify extreme scenarios, and thus, gain a deeper insight into opportunities and risks.

The next section summarizes the requirements for scenario planning support at the database, which were partially illustrated by the presented use cases.

1.3 Supporting Scenario Planning at the Database Level

As noted before, a range of analytic solutions provide functionalities to support calculations such as the ones that occur in the introduced use cases. Those solutions include statistical computing languages (such as R[1] and MatLab[2]), spreadsheet applications (such as Microsoft

[1]http://www.r-project.org/

[2]http://www.mathworks.de/products/matlab/

Excel) and customized spreadsheet-based solutions, as well as specialized tools for Enterprise Performance, Strategy, Risk and Financial Management (such as SAS Risk Management, SAS Financial Management [SAS10]). While those applications offer sophisticated functionalities, the need to transfer, integrate, and translate data between the data layer(s) and the application layer leads to a loss of efficiency, especially if large amounts of data are to be processed. The implementation of basic—often data-intensive—operators at the database can yield significantly faster run-times. Large data transports from the database to the application tier can be avoided. Further, moving basic computations to the database level can improve the maintenance of the provided functionality, both in terms of enhancements (newly added functionality is available to all applications accessing the database) and improvements to existing functions (optimization in terms of functionality and performance are available to all applications).

Those aspects have led to increased efforts both in research and industry to provide parts of the application logic for data-intensive applications close to the data, i.e., implement core functionalities for data representation and processing directly at the database layer [Pla09, JLF10]. Similarly, the database community has increasingly addressed the management of specific data characteristics, such as probabilistic, temporal, and scientific data, directly at the database such as to enable the consistent and efficient processing and management of such data.

Following this motivation, this thesis addresses the provision of functionality for uncertain data analysis and planning at the database. A data model for uncertain scenario data and a set of basic operators shall serve as a foundation to enable the creation, analysis, and modification of data in the scenario planning process.

The three central aspects addressed by the thesis are outlined in Figure 1.5. Those aspects are (i) the *data representation* and storage of uncertain data, (ii) their processing within an *analysis process* through a set of basic *operators*, including the flexible analysis of multivariate data under arbitrary correlation patterns, and (iii) the capture of scenario *provenance* and the analysis of change impacts by means of an efficient *recomputation* approach. Below, each of those aspects is described briefly.

1.3.1 Model and Analysis Functionality for Uncertain Scenario Data

As a foundation we require a data model that allows the representation of the different aspects of data used in the process of scenario planning. As depicted in the left-hand side of Figure 1.5, both historic fact data and uncertain data, reflecting assumptions or forecasts, can be subject to the process of scenario planning. They can be inputs, intermediate results, or outputs of the process. Uncertain data are represented through distributions over one or multiple variables.

Note that, in the remainder of the thesis, the working assumption is that the considered data is organized according to the multidimensional data model as described, e.g., in [CD97, Leh03]. This applies for historic data as well as for uncertain information. That is, the analyzed data is organized along so-called *dimensions* with (hierarchies of) dimension attributes (e.g., *day*, *month*, and *year* for the temporal dimension). The data is stored in fact tables, where values of relevant *key figures* are associated with values of the dimension attributes. Based on this model, one can apply typical Online Analytical Processing (OLAP) operations

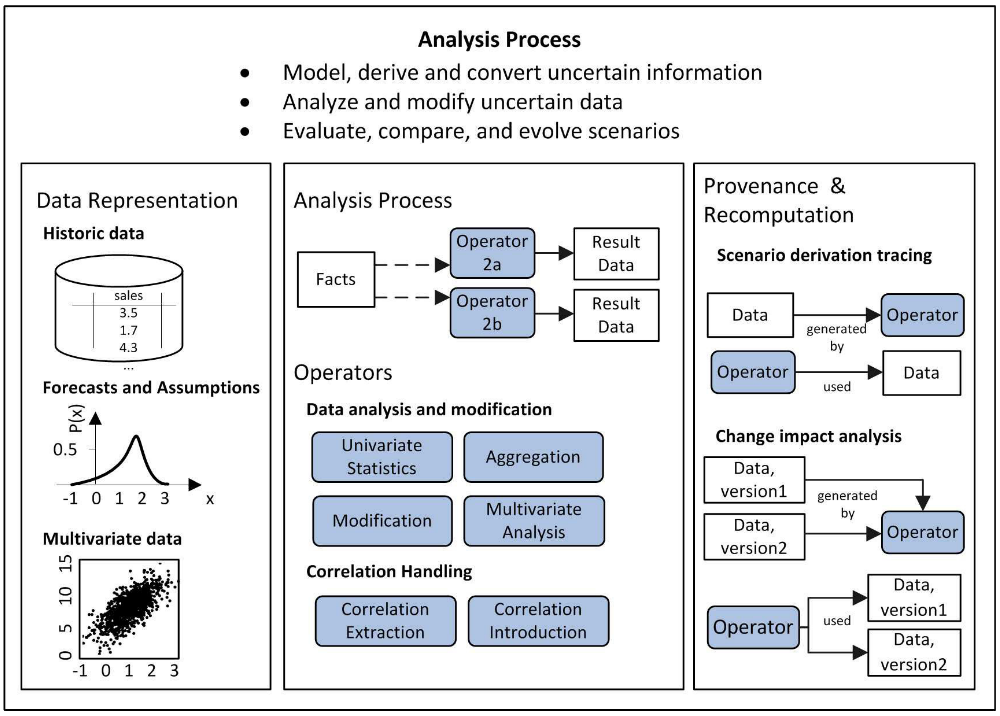


Figure 1.5: Central aspects of uncertain data in scenario planning

such as filtering and aggregating the fact data based on the dimension attribute values. When describing the analysis of uncertain values, we will consider their analysis according to the multidimensional model similar to certain data.

To address the special aspects of *uncertainty* in scenario data, we require support for

- Representing uncertainty in key figures (such as the prospected sales values from Use Case 1) and the dimensional association of data (such as the uncertain product launch times in Use Case 1) and

- Representing correlated uncertain data (i.e., multivariate distributions) reflecting the co-occurrence of multiple key figures (such as for the insurance claim amounts and rainfall intensity in Use Case 2)

Based on a suitably flexible data model, users further must be enabled to interact with the data. That is, they need means to create, analyze, and modify data with the aim to derive scenarios that reflect assumptions of interest. The thesis addresses this aspect by providing a set of operators processing both deterministic and uncertain data, which can be assembled in an analysis process as indicated in the center of Figure 1.5.

Modeling, Deriving, and Modifying Uncertain Information To enable the evaluation of different assumptions underlying a scenario, users need support for

- Constructing uncertain data flexibly, either by directly modeling distributions, or by deriving them from underlying fact data (e.g., an assumed and thus uncertain sales value may be derived from historic data, as in Use Case 1),
- Modifying uncertain values to create new or *adapt* existing scenarios. For example, as time progresses, a previously applied assumption regarding the expected sales can become increasingly inconsistent with actual data. It is necessary to reflect such deviations in the data.

Analyzing Uncertain Information A user needs means to analyze uncertain data similar to the analysis of certain data. That is, he requires functionality to access and analyze uncertain key figures, including dependencies between those key figures, and data with uncertain dimensional (especially, temporal) associations. To this end, it is necessary to provide a set of operators, such as

- Uncertain analogs to traditional relational operations such as selection and aggregation operators, which take into account the uncertainty of key figure values and the dimensional association of data,
- Basic operations for computing statistical features of univariate data, such as the moments of distributions, or the fit of distributions to given fact data, and
- Operators for multidimensional analysis.

It must be noted that the set of operators considered in the thesis is not—and can not be—a comprehensive set of functionalities for statistical computing and financial planning. Rather, a subset of operators has been chosen to enable the implementation of basic data analysis and modification steps that are relevant for scenario planning, as illustrated in the use cases. Regarding the analysis of multidimensional data, the flexible handling of correlations in data was identified as an important aspect in its own right and will therefore be addressed in depth, as outlined below.

1.3.2 Modeling, Extracting, and Evaluating Correlation in Data

To enable the evaluation of arbitrary correlation patterns in data and the analysis of multivariate distributions under such correlations, one requires means for

- Constructing, extracting, and storing correlation structures of arbitrary form and degree,
- Applying arbitrary correlation structures to arbitrarily distributed data, and
- Evaluating correlation structures in data efficiently, e.g., to compute risk measures.

The analysis of different correlation patterns that may potentially exist in data is an important means to enable an expert to gain insights into *possible* joint occurrences of events, as illustrated in the tasks for insurance risk evaluation in Use Case 2. In the thesis, we focus on the evaluation of complex correlation patterns in an ad-hoc fashion, and therefore, mainly address the handling of low-dimensional distributions. The analysis over higher-dimensional distributions requires alternative approaches. Those will be briefly discussed but otherwise not addressed in detail in this work.

1.3.3 Tracking the Provenance of Scenario Data and Evaluating Change Impacts

Besides the actual process of analysis and planning, it is essential that users can track back the results of a scenario analysis process to the base data and evaluate the assumptions that were made to derive a given result. This is required to enable a sound analysis (interpretation) of scenarios, the creation of new scenarios based on existing ones, and the recomputation of scenarios based on changes in base data. In this thesis the focus will be on the latter aspect, which is indicated in the right-hand side of Figure 1.5. To support an efficient recomputation of results, both the data model and the operators for analysis and modification of scenario data need to provide means for

- Capturing information about involved data items and processing steps applied in the analysis process,
- Accessing this information, and
- Evaluating the impact of changes in base (input) data on result (output) data in the process of recomputations.

On the conceptual level, keeping information about how data was derived generally is essential to enable users to understand, interpret, and possibly adjust a scenario. For example, when users arrive at a promising result for a given scenario that is associated with a high risk, they can track down the source of this risk. Then, they can get an insight to possible measures that prevent or counter the risk. Without knowledge about the derivation of a scenario, such an interpretation becomes impossible, since a user cannot understand the source of uncertainty.

1.4 Outline of the Thesis

The structure of this thesis is outlined in Figure 1.6. The major aspects will be covered as follows:

Chapter 2 provides the *foundations* concerning both conceptual aspects of *scenario planning and uncertainty* in data, as well as providing the reader with a technical foundation of *uncertain data management* and basic issues regarding the *provenance* of data. The previously introduced use cases will serve to exemplify selected aspects throughout the foundation, as well as for illustrating new concepts presented in the succeeding chapters.

In Chapter 3, we discuss concepts for *data representation and analysis* of uncertain data that build the basis of this thesis. In particular, we discuss the *representation of distributions*

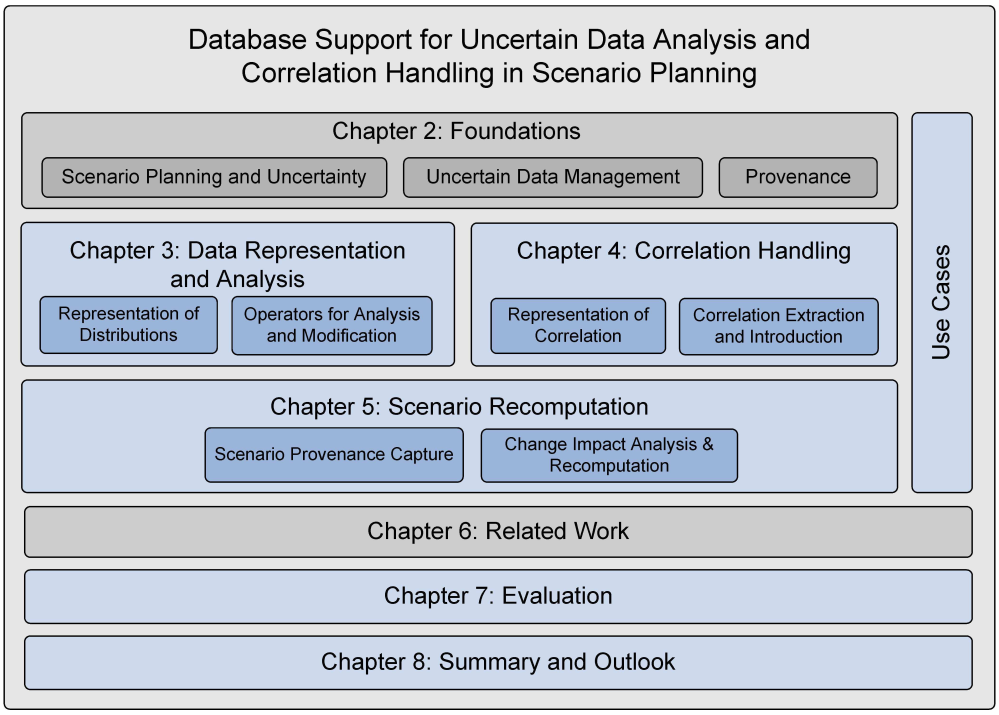

Database Support for Uncertain Data Analysis and Correlation Handling in Scenario Planning
Chapter 2: Foundations
Scenario Planning and Uncertainty
Uncertain Data Management
Provenance
Chapter 3: Data Representation and Analysis
Representation of Distributions
Operators for Analysis and Modification
Chapter 4: Correlation Handling
Representation of Correlation
Correlation Extraction and Introduction
Chapter 5: Scenario Recomputation
Scenario Provenance Capture
Change Impact Analysis & Recomputation
Use Cases
Chapter 6: Related Work
Chapter 7: Evaluation
Chapter 8: Summary and Outlook

Figure 1.6: Thesis outline

with focus on continuously distributed data, and provide a set of *operators for analysis and modification* of such data. Chapter 4 then focuses on the central aspect of *correlation handling* and describes a generic concept for the *representation of correlation* at the database as well as approaches for both the *extraction and introduction* of correlation structures to data, which build the basis for flexible multivariate analyses. Chapter 5 investigates the *recomputation* of analysis results. Specific focus is on the *capture of scenario provenance* information that can be exploited for an efficient *change impact analysis and recomputations* of analytical operators in the scenario planning process.

Chapter 6 discusses *related work* in the fields of uncertainty management and provenance, and gives an overview of solutions that apply those approaches in order to position them in the context of the approaches presented in this thesis. The experimental *evaluation* of the presented functionality is discussed in Chapter 7. The evaluation addresses the efficiency, the restrictions, and the optimization potential of the proposed concepts. Finally, Chapter 8 concludes the thesis, and provides an *outlook* on possible extensions and future research topics.

2 Foundations

This chapter introduces the reader to various concepts that form the foundation of this thesis. First, aspects of managing performance and risks in enterprises are introduced on a conceptual level. We situate scenario planning as an approach that can help improve risk management in planning processes within an enterprise performance management cycle. Uncertainty is a crucial issue in planning processes, as their true outcome is always unknown. We discuss how scenario planning is used to tackle the potentially complex uncertainty. Then, the topic of uncertainty—or, generally speaking, imperfection—in data is addressed and the technical aspects of uncertain data representation and processing in databases are discussed.

Finally, we turn to the related topic of data provenance, which describes how an artifact—such as intermediate or final data in a planning process—has come into existence. Provenance is crucial for managing and investigating scenario data and concerns both the process- and data-related aspect of our work.

2.1 Business Application Context

This section addresses the application context of the thesis from a business-centric point of view. We consider the concept of performance management and its relation to risk management and then discuss scenario planning as a means to address the management of risks.

2.1.1 Enterprise Performance and Risk

Enterprise Performance Management (EPM) is an integrated approach to increase the performance and profitability of a company. Similar to the traditional focus of Business Intelligence (BI), EPM comprises techniques, tools and processes to analyze and report developments of the business (see, for example, [Cok09, Bos06] for an overview of methodologies and techniques). EPM extends the usual historic focus of BI, by including also the *forecasting and planning* of future developments. The overall goal of EPM is to optimize relevant figures that reflect the performance of a company, such as its profitability.

Figure 2.1 displays a typical EPM cycle, including four stages that connect the different phases of execution in a business. Initially, one needs to develop a model that captures the historic and current state of the business as well as the potential future development of performance ①. By developing a strategy and prioritizing and breaking down the goals of a company, managers set concrete objectives, such as a figure for the return on investment which shall be achieved ②. The aim of planning is then to determine activities in such a way that the set objectives are most likely to be achieved ③. A plan may be re-organized during its execution, if required. During and after the execution of a plan, actual figures (e.g., the achieved revenue or product quality) are monitored, recorded, and compared to the planned figures in

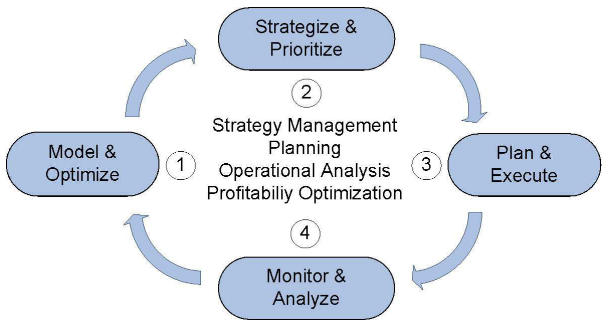


Figure 2.1: An Enterprise Performance Management cycle

order to evaluate the achievement of the plan ④. Finally, insights from this process serve to adjust the model or optimize the further execution of plans.

In the stage of planning, an expert applies models and business rules specifying information about drivers in the business—either defined explicitly or implicitly in the expert's mind—to project future values of relevant figures. For example, a planner may forecast a prospective sales quantity for a given product, using a forecast algorithm that exploits historic sales numbers and internal marketing information. Similarly, he may include prospected economic growth numbers in the analysis. At this point, it is important to recognize that the resulting sales figures will be based on *assumptions* regarding the considered drivers. Obviously, there is a chance that some assumptions will not hold true—or that the selected drivers are inappropriate or incomplete. For example, there might be an unexpected economic decline that renders the initial planning invalid. In brief, there is always a risk that wrong assumptions are applied in the planning process and thus expected results will not be achieved.

Risk There is no common definition of what constitutes a *risk*. The Risk Management Group of the Basel Committee on Banking Supervision has defined risk in terms of "risk of loss resulting from inadequate or failed processes, people and systems or from external events." [oBS01]. All definitions share the central idea that, first, risk is associated with an *event* that arrives with some *likeliness* (i.e., is not certain), and second, this event will have a *consequence* associated with an impact (mostly measured in terms of financial loss) on the performance of the organization. Figure 2.2 shows how the *likeliness* of the event and its potential *impact* (i.e., implied costs) both influence *risk*, which is computed as their product. One can consider both exogenous and endogenous risks. The former refers to risks such as the occurrence of a natural catastrophe or an economic decline. The latter refers to risks relating to the execution of internal processes.

Endogenous risks are risks that can be influenced (to a certain extent) by the decisions taken in an optimization. In contrast to that, the exogenous type cannot be mitigated. The informal definition of risk given above considers only the negative aspect of risk (where the occurrence of an event is associated with a negative impact). In scenario analyses, we also consider the positive interpretation of risk, i.e., risks associated with events that cause a *positive impact*.

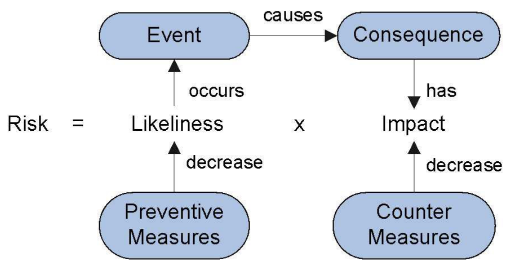


Figure 2.2: The relation of risk and measures in risk management

Therefore, besides trying to *minimize the risk* of a negative event, we also want to *maximize the chance* to increase performance by taking an opportunity.

Risk Management Managing risks inherent in the planning process has become increasingly important during the last decades. On the one hand, the high speed of change in the economic environment makes a more advanced risk management approach necessary in order to allow to react to changes more flexibly. On the other hand, legal obligations such as the Sarbanes-Oxley Act render risk management as a mandatory component in the management cycle of a company. The goal of a risk management process is to handle risk by taking measures that help (a) to minimize the likelihood of a loss occurring (*preventive measures*), or (b) to minimize their potential effect on the organization (*counter measures*). This relationship is illustrated in the lower part of Figure 2.2. An example for a preventive measure is to establish a quality assurance process to decrease the risk of low product quality. An example for a counter measure is to conclude an insurance that covers financial impacts of materializing risks.

Figure 2.3 shows a general approach to risk management. It comprises the identification of risks through appropriate tools and the identification and analysis of relevant root-causes (drivers) of those risks. A further stage addresses the assessment of risks by measuring their likelihood and the potential impact of the risk materializing. In the fourth stage, plans are devised to *mitigate* risks through proper measures that help reduce or counter the identified risks. Finally, the last stage of the risk management methodology considers the measurement and evaluation of identified risk indicators and drivers during the execution of processes, in order to identify plan deviations and enable counter measures at an early stage.

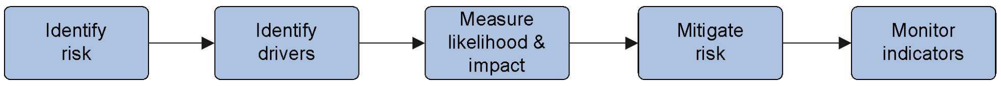


Figure 2.3: Stages of a generic risk management methodology

This work does not address the whole process of risk management nor does it approach risk management on a formal basis. Rather, we consider risk management as the context to be supported by representing uncertainty in data in a suitable fashion, and managing and processing such data through appropriate statistical functionality in an analysis process. One crucial factor to be kept in mind when addressing risk management is that one should also consider behavioral and subjective aspects in the process, especially so for the identification and evaluation of risks [Bie02, LKA03]. The technique of scenario planning addresses those human factors in the assessment of risks and chances.

2.1.2 Scenario Planning

The decision for a strategy or—on a lower level—concrete activities that influence the development of a company determine the company's exposure to risk. The strategy an organization will choose generally depends on its desired risk exposure in relation to the identified opportunities and risks associated with possible actions. For example, if a company is completely risk averse, they will choose activities such as to minimize the risk implied and thus devise plans so that it is very probable that a minimal target will be achieved. In the usual case, however, the goal is to find an optimal plan that balances risks and chances. To achieve such a plan and manage inherent risks it is not sufficient to produce traditional forecasts, since those fail to consider significant and unlikely changes. When considering only the *likely* outcomes of a process, both very high risks and opportunities may be completely overlooked. Therefore, it is valuable to enable the evaluation of potential developments under different assumptions concerning uncertain aspects. This is the heart of scenario planning.

Managing Uncertainty with Scenario Planning In the 1970's and 1980's, scenario planning was adapted from the field of military strategy planning as a tool for business planning, extending traditional planning approaches to take into account the growing uncertainty of future developments. The authors of [RS98] describe it as "that part of strategic planning which relates to the tools and technologies for managing the uncertainties of the future". Rather than forecasting *probable* future states or creating visions of *desirable* states, the focus of scenario planning is to explore *possible and plausible* futures by means of a well-defined number of scenarios. In many situations, the number of drivers that influence the future state of a business can be quite large (potentially infinite). The aim of scenario planning is to help to manage such complex uncertainties by focusing on a manageable number of scenarios, yet taking into account heterogeneous assumptions about the future [LB02].

If scenarios are well defined and do not include inconsistent assumptions, we consider each scenario in the set of derived scenarios as an "internally consistent view of what the future might turn out to be" [Por85]. That is, a scenario describes the state of the future given the applied assumptions are met. The described technique requires that a user first identifies the individual dimensions of the underlying uncertainty (detecting drivers) and second distinguishes a number of appropriate discrete alternatives as basis for alternative scenarios.

The central idea of scenario planning is sketched in Figure 2.4, which indicates two phases. The scenarios evolving from the first phase (Scenario Creation) effectively serve as an input to the second phase (Strategic Planning). The second phase constitutes the actual decision making, i.e., the derivation of business decisions based on an analysis of various possible scenarios. In the figure, the central characteristic of scenario planning is indicated through an

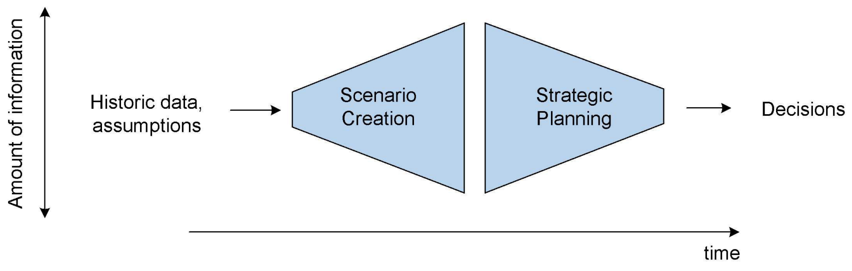


Figure 2.4: The two steps of scenario planning (adopted from [LB02])

evolving amount of information to be considered by a planner: First, one creates a certain number of scenarios (and thus, increases the information) by applying various assumptions to process available base data. Then, those scenarios are reduced in a second step to actionable information. This work focuses on database support for the first phase, by providing functionality that enables users to represent, derive, explore and subsequently manage and compare different scenarios. The second phase, including the actual decision making, is out of scope of this thesis, since domain-specific expertise and contextual knowledge is required to interpret different scenarios derived in the first phase. The same applies to the definition of specific risk functions, rules, or business models, which depend on a broad knowledge of the respective domain and are therefore similarly not considered in the thesis.

2.2 Uncertainty in Data

A planner's knowledge regarding future developments is always imperfect. Generally speaking, imperfect information is information that does not completely, not with certainty, or only inaccurately reflect the true state of the world. The consequence of imperfection is uncertainty in the data, i.e., a "lack of certainty" whether some proposition holds true, or how exactly it describes the real world. In this section, we first address different types and sources of uncertainty. At this stage, we consider uncertainty with respect to a generic data item, be it a single value or tuple in a database, or an event in the general sense. Where appropriate, we clarify specific features through an example.

2.2.1 Kinds of Uncertainty

There are a number of types, or possible interpretations, of uncertainty. Here, we address *imprecision*, *incompleteness*, and *confidence* of information as possible forms of uncertainty. However, there is no *one* clear classification of uncertainty and thus, those types neither constitute an exhaustive list nor do they unambiguously partition the space of uncertain information. Often, one could apply different interpretations of uncertainty in the same context, and vice versa, varying contexts can imply varying interpretations of uncertainty for the same data.

Imprecision of data can occur when data values cannot be provided precisely or when information is not available at the finest level of granularity represented in a database.

Such imprecision can be expressed through ranges (e.g., a *sales quantity* will be between $10k$ and $20k$), or through a set of possible values (e.g., the *sales quantity* will be $10k$, $11k$, or $12k$). Similarly, if one wants to query data at a lower level of granularity than the recorded granularity, there arises uncertainty in the form of imprecision. For example, when sales data was collected at the city level but shall be evaluated on the level of individual stores, one cannot say with certainty in which exact store a given sales amount was achieved.

A special aspect of imprecision is *indeterminacy* of data. Indeterminacy appears especially when addressing uncertainty of facts associated with a future development which is not determinate. Similar to previous literature [DS98, CP00] we use the term indeterminacy particularly to refer to *temporal* uncertainty. Indeterminacy is relevant when the occurrence and duration times of events are not exactly known. For example, the launch date of a product may not be known precisely due to an unpredictable development phase. Furthermore, there are cases where we want to express an explicit "tolerance" with respect to a temporal value. Consider as an example a customer's requirement that an order shall be delivered within the next two to four weeks. When planning the delivery, the possible occurrence time span should be taken into account, rather than an exact point in time.

Incompleteness of data often results from technological constraints or intrinsic characteristics of a data acquisition process. For example, in production environments, measurements might be missing due to defect sensors. Similarly, when conducting customer surveys, customers might not answer all questions, e.g., regarding their salary or expenditures, leading to incomplete data. Incompleteness can be regarded as an extreme form of data imprecision in the sense that a missing value could potentially take on any value from the domain of possible values, e.g., the $salary$ of an interviewed customer may be arbitrarily high.

Confidence describes the *belief* in the validity of a given data item, or the likeliness of an event. An example in the above context is the statement that the amount of products sold next year will very likely be $10k$, or, conversely, will very unlikely be $100k$. One can express the degree of confidence through a numeric confidence factor, such as the probability (see Section 2.3.1) that the given data item reflects reality. As stated in [Mot96], in many cases the confidence and precision of a data item can be traded off against each other in the sense that a higher precision of a data item (restricting it to a smaller range of possible values) implies that the confidence in the data decreases, and vice versa.

The aspects of *imprecision* and *confidence* are the most relevant factors when dealing with data which is forecast or approximated from historic data (with imprecision) or reflects an expert's estimations (with indeterminacy and confidence). Therefore, in this thesis, the focus is primarily on uncertainty in terms of imprecision and confidence.

The remainder of this section focuses on different aspects concerning the granularity and source of uncertain data.

2.2.2 Granularity of Uncertainty

In the context of scenario planning, uncertain data is either the input or result of a data analysis or transformation step. The uncertain information can exist on different levels of data granularity. One typically discerns uncertainty on attribute level, tuple level, and relation level. For example, the future value of an attribute $sales\ quantity$ for a product may be known only imprecisely, e.g., to be in the range $[2k, 3k]$ or to be either $2k$ or $3k$. Analogously, uncertainty can be on the level of tuples, e.g., we may not know for certain whether a sale was made, or we may be uncertain whether a specific $product$ was sold in $storeA$ or $storeB$. In the latter case, both the tuples $(product, storeA)$ and $(product, storeB)$ are considered uncertain tuples.

One can also associate uncertainty with a complete relation, i.e., a whole set of tuples that may or may not reflect the true state of the world.

In this work, the focus will be primarily on uncertainty on the level of (numeric) attributes. This is because scenarios are mostly derived from alternative assumptions regarding *key figures* and *measure values* that are relevant to analyze a business aspect of interest. Still, the concepts of tuple- and relation-level uncertainty remain relevant, for example, when we want to express the confidence in a complete derived scenario.

2.2.3 Sources of Uncertainty

Similar to the different interpretations of the concept of uncertainty, we can also discern different sources from which uncertainty can arise or reasons why uncertainty is present in data.

Uncertainty in Base Data Many types of *base data* from the "real world" are inherently uncertain. This applies, e.g., to trend data derived by a forecast mechanism, sensor measurements, or the results of information extraction mechanisms. Of course, in those cases the uncertainty has been initially introduced to the data at some point in time (during data acquisition). For example, in the case of sensor data, measurements are always imprecise, where the degree of imprecision depends on the sensor quality. Here, the term *base data* refers to the state of data at the point of *insertion to the database*. An example for uncertain base data that is relevant for scenario planning is a forecast value acquired from an external service.

Uncertainty Introduced Through Processing Uncertainty can naturally also be added to data through applied transformations (queries, modifications, or computations for deriving new data). Often, processing steps—particularly in the context of scenario planning—include an assumption and thus an inherent uncertain component that is introduced to the data. Further, uncertainty may be introduced through indeterministic processing steps, e.g., when applying sampling techniques in the implementation of a processing step.

Uncertainty About the Uncertainty On a meta-level, it is important to be aware that any quantification of uncertainty is itself uncertain. That is, we cannot determine with absolute certainty whether a determined degree of belief reflects the "true" certainty of the represented information. For example, in information extraction processes, different heuristics may lead to varying confidence values for the same entity, rendering the confidence values uncertain. In the context of scenario planning, one must consider the restriction imposed through the human factor: The quality and coverage of derived scenarios will naturally depend on the user's experience, expertise, as well as their willingness to investigate scenarios that do not reflect the most likely (or most preferred) developments. While this factor should always be considered when evaluating the practical application of scenarios, it is not further discussed in this work.

Uncertainty About the Considered Data Schema Another kind of meta-level uncertainty stems from a lack of information about the suitable or preferred schema that should be used for modeling the data at hand. This aspect is addressed, e.g., in the context of so-called *malleable schemas* [DH05] which incorporate imprecise and overlapping definitions of data structures in data integration processes.

For this thesis, the first three sources of uncertainty are relevant. First, we consider uncertainty in data that are already represented in the database or imported from external sources. Second, as processing steps are executed, they often introduce some assumption and thus uncertainty to the data. The aspect of meta-level uncertainty is also a relevant facet since it must be considered when interpreting the associated uncertainty of alternative scenarios. The interpretation of uncertainty in concrete application contexts or business domains are highly complex topics in their own right. Those aspects are out of the scope of this work.

2.3 Representing Uncertain Data

This section discusses selected aspects of uncertain data representation and points out the most relevant issues in the context of our work. Particularly, we consider the distinction of different levels of representing uncertain data, their underlying distribution characteristics, the explicit and implicit representation of uncertainty, and approaches for representing correlation (dependency) in data.

While there exists a broad range of approaches for modeling uncertain information, the goal of this section is not to provide an exhaustive survey of models and systems. Relevant related approaches for uncertain data management are discussed later on in Chapter 6. A broad survey covering models, properties, and querying algorithms for uncertain data is given in the work of [SBH$^+$09]. It provides a generic model for uncertain data representation, which can be specialized to concrete models for various application cases. The authors of [SBH$^+$09] focus on traditional uncertain data management, enabling the representation of and relational queries over uncertain data. Further classes of transformations, such as statistic analytic functions, are not considered. In the following sections, we will refer to the concepts applied in [SBH$^+$09] where appropriate, while additionally pointing to aspects not covered by their model yet relevant for the present work.

First, we consider the semantics of *belief* or certainty in a data item formally, addressing probability theory and possibility theory as two alternatives for quantifying uncertainty.

2.3.1 Probability and Possibility of Information

The degree of uncertainty of a data item can be represented through confidence values. One can use different such values, depending on the application context and the applied theory. For example, one can assign categorical belief values such as “very uncertain”, “relatively certain” or “very certain” to describe the belief in a given data item such as a forecast value. Mostly however, one uses numeric confidence values in the range $[0, 1]$, with 0 representing a complete lack of confidence, and 1 representing complete certainty. Two alternative concepts to deal with degrees of certainty are *probability* and *possibility*.

Probability Theory and Possible Worlds Semantics

Under probability theory, uncertain information of a data item is represented through a random variable x which is associated with a probability distribution P_x over the possible instantiations of x, i.e., the domain of x. The probabilities of all possible instantiations sum up to 1, meaning that the variable will take *one* of those values with certainty.

Probabilistic Databases Most approaches for uncertain data management assume the probabilistic representation of uncertainty. Tuples and attribute values are assigned probabilities of belonging to the database, according to an underlying probability distribution. A possible instance of such a database, termed *possible world (PW)* can be constructed from any (valid) combination of tuples in the database. A *probabilistic database (PDB)* is then defined as a probability distribution over the set W of all possible worlds. Each $w_j \in W$ specifies an assignment $\models_{j\ x \leftarrow v}$ of a value $v \in dom(x)$ to each random variable x contained in the database according to the distribution P_x, with $\sum_{w_j \in W} P(w_j) = 1$. That is, in each possible world all uncertain attributes (tuples) are assigned a deterministic value (tuple), leading to a deterministic database.

As the database represents a probability distribution over possible worlds, the result of queries over this database will also be a probability distribution. That is, each tuple (or attribute value) contained in a query result is associated with a result probability. This is termed *Possible World Semantics (PWS)*. Under PWS, the probability $P(x = v)$, i.e., the probability that an uncertain variable x has value v, is computed from the individual probabilities of all possible worlds w_j where $x = v$, i.e., $P(x = v) = \sum_{w_j \models_{x \leftarrow v}} P(w_j)$.

When random variables are associated with continuous distributions, which is the assumption underlying the approaches of this work, W would contain an infinite number of possible worlds. Therefore, a representation and evaluation of continuously distributed data under PWS is not feasible. Section 2.3.4 will address possible representations of continuous random variables, while Section 2.4 discusses basic techniques that underly the processing of such variables in this work.

Possibility Theory

The concept of possibility theory is an alternative to probability theory based on fuzzy sets. A fuzzy set is a set of possible values, where each value is assigned a number in the interval

$[0, 1]$ indicating the possibility that the value is a member of the set. In contrast to probabilistic semantics, the sum over all the assigned degrees need not be 1. A possibility distribution Π_x defines a unique set of probability distributions that are consistent with Π_x. Fuzzy sets and possibility theory build the basis for *fuzzy database models* [Ma06, BK96, YG98], where uncertain values are associated with possibility distributions.

Overall, possibility theory is applied to a larger degree in the field of artificial intelligence rather than in the area of uncertain databases. In this thesis, fuzzy data modeling will not be considered further.

2.3.2 Granularity of the Representation

Information about uncertainty is represented on different levels depending on the granularity of the considered data items. We discern attribute-level, tuple-level, and relation-level uncertainty.

Uncertainty on Attribute Level Attribute-level uncertainty addresses incomplete information about the concrete value of an attribute value, i.e., uncertainty within a field of a tuple. For example, for a relation $Sales(product, location, quantity)$, we may be certain that a sale was (will be) made, but we may be uncertain about the concrete sales $quantity$; for example, we may not know for sure whether the customer orders $2k$ or $3k$ of a given product $product1$.

Uncertainty on Tuple Level Tuple-level uncertainty addresses the uncertainty about the existence of tuples in the database. For example, for the relation $Sales(product, location, quantity)$ it may not be known exactly whether a sale $s1(product1, cityA, 2k)$ was (will be) made, and therefore it is uncertain whether tuple $s1$ exists. Such an uncertain tuple is associated with a random boolean variable x which indicates the tuple existence. For a specific instantiation of the probabilistic database, the tuple exists if x takes value 1; otherwise, the tuple does not exist.

In terms of [SBH$^+$09] tuple-level and attribute-level uncertainty are defined using the concept of *A-tuples*. For each attribute in a relation, an *A-tuple* contains either a deterministic value or an *OR-set* of possible attribute values. A specific *A-tuple instance* is defined as an A-tuple for which a concrete value has been assigned to each attribute. This concept can be applied to represent various models of uncertain data, as shown in the survey. The representation of attribute uncertainty is implicit in the definition of OR-sets, since those specify sets of possible values for A-tuple attributes. For example, when the true value of $quantity$ is known only to be either $2k$ or $3k$, we can represent this value by an OR-set $\{2k, 3k\}$, as shown in Table 2.1 next to another tuple $s2$, whose $quantity$ value is certain. All valid combinations of value assignments to all attributes of an A-tuple then represent the alternative tuple instances for that specific A-tuple. In the example, this results in two alternative tuple instances $(product1, cityA, 2k)$ and $(product1, cityA, 3k)$.

Apart from specifying alternative attribute values, there may be a confidence attached to each of the alternatives in an *OR-set*, which describes the probability that the corresponding

Table 2.1: The $quantity$ value of tuple $s1$ is defined by an OR-set

id	product	location	quantity
$s1$	$product1$	$cityA$	$\{2k, 3k\}$
$s2$	$product2$	$cityB$	$1k$

A-tuple instance exists, as shown in Table 2.2. Note that the probabilities of the two alternative A-tuple instances $(product1, cityA, 2k)$ and $(product1, cityA, 3k)$ sum up to 1, as is required under PWS.

Table 2.2: The alternative $quantity$ values occur with different probability

id	product	location	quantity
$s1$	$product1$	$cityA$	$\{2k : 0.2, 3k : 0.8\}$
$s2$	$product2$	$cityB$	$1k$

In this work we consider uncertainty mainly at the level of attribute values. The analysis of business scenario data mostly is based on computations over numeric measure attribute values, describing some quantity, rather than considering the (non-)existence of individual facts. Therefore, the attribute-level representation is most natural in the context of analytic evaluations of business-related data. For example, when prospecting sales for a selected location, it is assumed that *some* sales quantity will be achieved for each product, while the concrete quantity may well be uncertain. Thus, it is natural to represent the value of the $quantity$ attribute as an uncertain value.

Uncertainty on Relation Level Table 2.1 displays a simple uncertain relation. Recall that, under PWS, a possible instance of an uncertain relation is a concrete (valid) assignment of a deterministic value to *each* uncertain tuple in the relation. That is, an uncertain relation represents a distribution over all possible (valid) combinations of tuples. In terms of [SBH$^+$09] this corresponds to all combinations of possible *A-tuple instances* in the relation. Thus, for the given example, we have two possible instances of the uncertain relation $Sales$, as shown in Tables 2.3 and 2.4. If we consider the probability values given in Table 2.2, the probability that $Sales_1$ represents the true state is 0.2, while the probability of $Sales_2$ is 0.8.

Table 2.3: Possible relation $Sales_1$

id	product	location	quantity
$s1$	$product1$	$cityA$	$2k$
$s2$	$product2$	$cityB$	$1k$

Table 2.4: Possible relation $Sales_2$

id	product	location	quantity
$s1$	$product1$	$cityA$	$3k$
$s2$	$product2$	$cityB$	$1k$

So far, we have not explicitly considered the concept of *dependency* between data items. However, the definition of OR-sets implicitly contains information about dependency, as they specify sets of *alternative* attribute values (and correspondingly, of alternative tuple instances). This means that only *one* of the attribute values (and corresponding tuple instances) may exist in an instance of the database at the same time.

If variables are considered independent from each other, this implies that each combination of attribute values and uncertain tuple instances can occur. Moreover, the assumption of independence implies that the likelihood of two specific tuple alternatives occurring jointly can be computed as the product of their individual tuple confidences. However, this assumption is a strong restriction that mostly cannot be applied when considering real-world data. When dependencies exist between tuples, this determines how they can appear together in a relation, and thus influences the distribution of possible relation instances of an uncertain relation. Dependency, or correlation, is an important aspect that takes many different forms in real-world data. It will therefore be addressed in more detail in Section 2.3.5, and constitutes the central topic of Chapter 4.

2.3.3 Explicit and Implicit Representation

In the examples shown above, the representation of uncertainty is accomplished through the modeling of tuple and value alternatives and associated confidence values. This corresponds to an *explicit* form of representation, where the different artifacts to represent uncertainty are an integral part of the data model in the sense that basic relations are *extended* with additional attributes to store information that describes the uncertainty. As an alternative to the model extension approach, uncertainty can be represented in an implicit fashion, exploiting Monte-Carlo techniques (see, e.g., [JXW$^+$08]). Between those two approaches there exist major differences both regarding the representation of uncertainty and query evaluation over the represented data.

Explicit (Model Extension) Approaches Most approaches to uncertain data management rely on an explicit representation of uncertainty, where relation schemata are extended with information about the uncertainty. For example, relations may be extended with special attributes that mark optional or alternative values, or through an attribute indicating the existence probability of each tuple in the relation. Irrespective of the concrete model implementation, the use of model extension approaches implies that the types of representable uncertainty are hard-wired into the system. Thus, one must adapt the model when new types of uncertainty shall be supported, thereby potentially affecting dependent applications.

Implicit (Sampling-based) Approaches An alternative approach to modeling and processing uncertain data is to represent distributions of variables symbolically, and realize their concrete instantiation through *sampling* at runtime. The essence of this sampling-based approach is the use of Monte Carlo (MC) algorithms, which are applied in many fields such as physics, computer science, finance, telecommunications and games. The MC approach proceeds by sampling a high number of realizations for each random variable in a query through appropriate sampling functions. Then, those samples are evaluated deterministically, i.e., by evaluating standard relational operators over the sample sets and deriving an empirical result distribution. The great benefit provided by this approach is its genericity. Since arbitrary sampling functions can be applied, the class of applicable probability distributions is not restricted and not hard-wired into the data model.

Integration of Approaches Many systems combine model extension approaches with MC techniques. They aim to exploit the genericity of the latter for tasks which otherwise would be computationally too expensive or would not be solvable analytically. The concepts presented in this thesis rely mainly on the model extension approach. Sampling will be applied as a means to construct and apply correlation structures, as described in Chapter 4.

2.3.4 Distribution of a Variable

As exemplified above, probability values are used to reflect the confidence that a tuple or value alternative belongs to a possible database instance. Thus, uncertain *tuples* are associated with a random boolean variable X, which takes a value of 1 with a given probability (meaning t exists) or 0 (meaning t does not exist in the database). In the case of uncertain *attribute values* are similarly represented through random variables, associated with a probability distribution. This distribution can be defined over a discrete or continuous domain $dom(X)$.

Discrete Distributions In the discrete case, X takes a value out of a *finite number* of possible values as characterized by a *probability mass function (PMF)* $p(x)$, which reflects the probability that $X = x$, with $\sum_j p(x) = 1$. The associated *cumulative distribution function (CDF)* $P(x)$ of such a discrete variable increases discontinuously over $dom(X)$. As an example, one can model alternatives for next year's economic growth being 1%, 1.5%, or 2% with a probability of 0.25, 0.5, and 0.25, respectively. See Figure 2.5 for an example of the PMF $p(x)$ and the CDF $P(x)$ in this case.

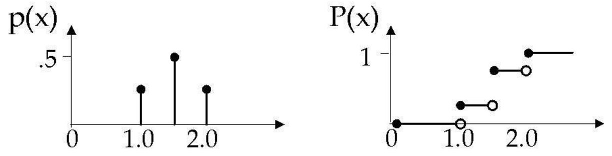


Figure 2.5: Functions $p(x)$ and $P(x)$ of a discretely distributed variable

Most approaches that provide for the representation of attribute-level uncertainty over discrete distributions have also added support for continuous distributions, addressing requirements from various application domains where the continuous form is a more natural representation.

Continuous distributions A continuous random variable X is characterized through a non-negative integrable probability density function (PDF), $p(x)$, and a continuous CDF,

$P(x)$. The density function $p(x)$ describes the likelihood of X occurring at a given point x[1], while the cumulative distribution $P(x) = \int_{-\infty}^{x} p(x)dx$ describes the likelihood that X will take a value $X \leq x$, where $\int_{-\infty}^{+\infty} p(x)dx = 1$. See Figure 2.6 for an example of the PDF $p(x)$ and the CDF $P(x)$ in the case of a normally distributed variable.

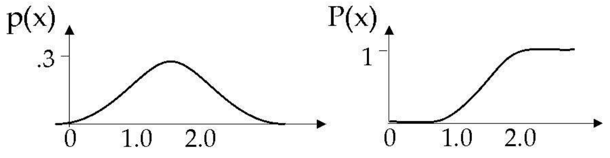


Figure 2.6: Functions $p(x)$ and $P(x)$ of a continuously distributed variable

As the number of possible values of a continuous random variable is infinite, the number of possible worlds represented by a probabilistic database which contains such variables is infinite as well. Thus, exact query computation over continuous random variables in terms of PWS is not possible, i.e., would yield infinitely large computation costs. Based on the specific distribution associated with a continuous variable, different representation forms are used to represent and process the associated distributions. A selection of those representations is listed below.

- **Parametric representations** represent the characteristics of known distributions in a compact form where this is possible. For example, one can represent variables following a Uniform, Normal, or Gamma distribution through the distribution family and appropriate parameters as *Uniform(min, max)*, *Normal*(μ, σ), and $Gamma(\alpha, \beta)$, respectively. For example, a symbolic representation of a normally distributed variable reflecting next year's economic growth value could be specified as *Normal*$(\mu = 1.5\%, \sigma = 0.5\%)$ (see Figure 2.6).

- **Characteristic functions** can be used to represent known distributions, allowing the system or a user to access specific features of the distribution efficiently. Examples include information about the *inverse cumulative distribution function (ICDF)* of a variable, or providing access to the probability $P[min \leq x \leq max]$ of X being within an interval $[min, max]$ by using a mass function $mass(min, max)$ (see, e.g., [ABS$^+$06, KK10]).

- **Histogram-based representations** approximate underlying distributions by reflecting the density within subintervals (bins) of the domain. Histogram-based approximations are used when the distributions cannot be represented symbolically or cannot be evaluated in closed form. The most basic form of histograms is the class of equi-width histograms. They partition the domain into equally-sized bins, each of which is associated with a frequency that reflects the mass of the distribution that lies in the bin. Equi-width histograms are constructed and updated very efficiently, but can yield large

[1]More precisely, $p(x)$ describes the likelihood of X occurring within a small interval $[x - \epsilon, x + \epsilon]$, $\epsilon \leftarrow 0$, around x.

approximation errors depending on the nature of the approximated distribution (e.g., for very 'spiked' distributions). More complex partitioning schemes serve to approximate distributions more accurately, yet at higher costs for both the initial construction and the update of the histograms, as discussed, e.g., in the work of [JPK^{+}98]. Examples include equi-depth histograms, which partition the domain into bins that cover an equal amount of the mass of the distribution, and V-Optimal histograms, which partition the domain such that the cumulative weighted variance of the bins is minimized. In this work, equi-width histograms are used due to their efficiency for both constructing and updating uni- and multivariate distribution approximations. More complex partitioning schemes could, however, be used as an alternative implementation for most of the presented approaches.

- **Monte Carlo approaches** rely on drawing pseudo-random samples over a continuous domain to implicitly model continuous distributions, as described briefly in Section 2.3.3.

2.3.5 Dependencies in Data

Dependencies in data are a highly important aspect for enabling consistent analyses over probabilistic data. In particular, investigating the effect of data dependencies on business developments can help experts to prevent misestimations of risks and chances, as indicated in the description of Use Case 2 in Section 1.2.2. Correlation between random variables, such as the losses of related insurance assets, is reflected in the joint distribution of those variables. Similar to the representation and processing of univariate probability distributions, we can distinguish two basic forms of representing correlations in data.

Correlation information can be represented in extended schemata where information about dependencies (e.g., implication, mutual exclusion, or degrees of positive or negative correlation) is stored explicitly. To this end, one can use additional attributes storing dependency information for each tuple, or record this information in dedicated tables. Noteworthy, many approaches consider the base data as such independent and focus on dependencies that are introduced through query processing (e.g., when joining data). This relates to the tracking of provenance, or *lineage* of data, which will be further addressed in Section 2.5.

Dependencies in Discrete Uncertain Data Current approaches to handling dependency in data mostly focus on the processing of dependencies between discrete random variables, i.e., between tuple alternatives and alternative attribute values.

The common underlying approach is to define constraints over random variables that represent dependencies between uncertain attribute values, or dependencies on the tuple-level. The concept of *probabilistic c-tables* [GT06] is an instance of this approach. A probabilistic c-table is a relational table which keeps a *local condition* for each tuple in the relation. The conditions specify how different valuations of variables can occur together, and hence, describe the dependencies between those variables. A possible world over c-tables is then associated with a particular value assignment to all variables that fulfills all conditions. Another representation form, which is equally expressive as probabilistic c-tables [AJKO08], is the representation of the dependency by means of probabilistic graphical models. Those represent joint distributions of a set of variables through a graph whose nodes represent the variables and whose edges represent direct dependencies between the variables. Both those approaches and their application will be discussed in depth in Chapter 6.

Dependencies Between Continuous Distributions In contrast to dependencies between discrete values and tuples, the specification and processing of correlation over continuous distributions has been addressed only to a limited extent in probabilistic data management so far. Yet, evaluating multivariate distributions over continuous variables, and investigating potential correlation structures in data is a highly important aspect, particularly in the context of scenario planning.

While there exist approaches (such as [SMS$^+$08, JXW$^+$08, KK10]) that generally allow for symbolical representation and processing of multidimensional distributions, they do not explicitly handle correlation information as a self-contained construct. This makes a concise evaluation of correlation structures a difficult task. Also, as stated before, *sampling-based* approaches generally allow us to represent arbitrary dependencies over continuous distributions (using arbitrary marginals and correlation structures). However, they similarly do not provide direct access to the *correlation structure* as such. Rather, the information about dependencies is encapsulated in the specific sampling functions. Thus, the comparison of correlation structures and the introduction of new correlation information to hitherto uncorrelated data (e.g., forecasts) can become a complicated task.

In the field of financial and insurance mathematics, the modeling of complex correlations plays an important role. Here, so-called *copula functions* are used as a well-founded statistical approach. A copula (Latin for "link") essentially is a distribution that represents the relation between individual distributions and their joint multidimensional distribution in a normalized fashion (for a detailed introduction, see, e.g., [Nel06]). The most important benefit of copulas is that they are independent from specific marginal distributions and invariant to linear transformations.

Sections 4.2 and 4.3 will discuss the concept of copulas in depth and introduce approaches to apply copulas at the database level for a flexible handling of arbitrary correlations over continuous domains.

2.3.6 Temporal Indeterminacy

In most of the examples introduced so far, we addressed primarily the representation of uncertainty in measure values, such as a sales quantity or the economic growth. Another important aspect in the context of planning is uncertainty in the dimensional data associated with an analysis, particularly, in the temporal dimension. In database research, the representation and querying of temporal information has been addressed both through the concept of interval-based temporal data (where the temporal occurrence of an event or fact is known to cover an interval in time), and uncertainty or *indeterminacy* in temporal information. It is possible to extend approaches for general uncertainty representation to represent temporal indeterminacy. Generally, the handling of temporal indeterminacy combines aspects from the domains of data uncertainty and temporal databases. In the following, we briefly address the basic representation of temporal data and temporal indeterminacy in particular.

Temporal Information Most applications of databases involve temporal data, since the management of real-world data usually involves some temporal aspect. Data is called temporal data if it is in any way connected to a time interval and this connection implies some special aspect for the data. The area of temporal databases has received broad attention during the last two decades due to the growing need to represent and manage temporally evolving data directly at the database, rather than addressing those issues at the application level. An important aspect addressed in temporal data management is the *validity* of facts within time. For example, data describing the association of an employee with a company might only be valid during a specific time interval $[t_1, t_2]$. To analyze and predict developments, e.g., of salaries or sales amounts, along time, one requires means to represent and query the temporal information. A comprehensive introduction to temporal data management can be found, e.g., in [JS99]. In the following, we introduce selected aspects that are relevant in our context.

The basic concepts for representing time and temporal validity in databases are *instants* and *durations*. An instant or duration is specified through a value of a suitable unit of time. This unit for temporal measurements is generically referred to as a *chronon*. Depending on the database system and application context, a *chronon* refers to the smallest unit of time representable, e.g., a minute, an hour, or a day.

Certain, i.e., *determinate instants*, denoted by t, can be represented through a single chronon. Likewise, *determinate intervals* $\underline{T}$ are considered periods in time and can be modeled through a lower and upper determinate instant that represent the boundaries of the interval, i.e., a start time t_s and end time t_e, such that $\underline{T} = [t_s, t_e]$.

The seminal work of Allen [All83] describes the representation of temporal intervals and their relations by means of an *interval algebra* . The interval algebra was introduced as a calculus for temporal reasoning, with primary focus on the field of artificial intelligence. It includes thirteen possible relations between intervals, such as that an interval *takes place before* or *overlaps* with a second interval, and provides algorithms for reasoning over temporal interval relations by means of a composition table. Allen's interval algebra can be considered the foundation for a broad variety of further research on temporal data representation and reasoning. While reasoning over temporal data is out of the scope of this work, the $overlaps$ relation plays an important role in the computation of aggregates over interval-based and uncertain temporal data.

Indeterminacy We address uncertainty in temporal data by the term *indeterminacy* as in [DS98]. The basic constructs of valid-time indeterminacy are *indeterminate time instants, periods, and intervals*. *Indeterminate instants* are represented by means of a lower and an upper support, t_l and t_u, and a probability distribution function that reflects the probability that the instant occurs at any of the chronons in the interval $[t_l, t_u]$. *Indeterminate intervals* are likewise represented by a set of possible durations and a respective probability function over those durations. Finally, *indeterminate periods* are represented through an indeterminate start and an indeterminate end instant.

In the context of this thesis the term of *indeterminate events* refers to facts in a database (reflecting planning-relevant events) that have an indeterminate temporal association. That is, indeterminate events are considered under interval semantics, i.e., as having an indeterminate start time and an indeterminate duration, which (implicitly) also determines their indeterminate end time. This approach is similar to the concept of *indeterminate periods* in [DS98]. As an example for indeterminate events, consider the indeterminate product launch times, and the associated indeterminate periods of sale in Use Case 1. The representation of temporally indeterminate event data and the aggregation of data associated with indeterminate events will be addressed in Sections 3.2.3 and 3.3.4.

In this section, we discussed different aspects of representing uncertain data. The following section describes approaches for processing such data during query evaluation.

2.4 Processing Uncertain Data

The two basic approaches to uncertain data representation (implicit and explicit) imply also different forms of query evaluation. In the case of sampling-based systems (implicit uncertainty representation), queries are evaluated over a large set of generated PWs in a deterministic fashion. In this section, we will focus on the processing of *explicitly* represented uncertain data with *continuous distributions*. We particularly address selected aspects of analyses over such distributions, including the computation of statistical measures, aggregation, and the handling of correlations in data.

2.4.1 Analysis over Continuous Distributions

Section 2.3.4 described different concepts to represent continuous distributions, as a basis to access their features during query processing. On the one hand, symbolic representations specify distribution information in a concise form and enable exact analytic computations in some cases. On the other hand, in practice only few distribution functions (such as the Normal distribution) allow an *efficient* analytical evaluation. Therefore, one resorts to approximate processing approaches in most cases, relying on one of the previously described representation forms.

In the context of planning, analysis functions over continuous data are a central requirement. Examples include the computation of statistical measures over uni- and multivariate distributions (see Section 1.2.2), such as the moments or quantile values of a distribution, or the probability that a value exceeds a given threshold (tail probability). Further, one needs functionality to filter data based on probabilistic thresholds, and to compute aggregates and formulas over uncertain values. Below, we consider some basic operations over (continuous) probabilistic data, as well as addressing their application in statistical computations and probabilistic database systems.

Integration and Numerical Approximation

Integration over continuous distributions is a central operation that builds the basis for many more complex functionalities. Given an interval $[a, b]$ and a PDF $p(x)$, the integral $\int_a^b p(x)dx$ computes the probability that x will take a value in $[a, b]$. Integration is, for example, required as a central functionality to evaluate probabilistic predicates or joins over uncertain values.

Also, histograms that are used as approximate representations of symbolic distributions essentially reflect the result of a piecewise (bin-wise) integral over the underlying PDFs.

If the distribution of x is represented symbolically, one may compute the exact integral analytically. In cases where an exact analytical computation is too expensive or even impossible, one can take an approximate, numerical approach to integration.

Numerical Integration One can compute an approximate integral over a function $p(x)$ by evaluating $p(x)$ at a number of discrete points within the integration interval, and computing a weighted sum of the partial results. Different integration rules, or *quadrature rules*, can be applied to this end. The simplest form is the so-called *rectangle rule*, which approximates $p(x)$ in $[a, b]$ by a constant function through the point $((a+b)/2, f((a+b)/2))$. By splitting up $[a, b]$ in equal-sized subintervals and adding up the results for all subintervals, one can increase the accuracy of the overall integration result to a desired degree. Another quadrature rule is the *Simpsons rule*, which approximates each subinterval through a second-order polynomial, and thus can serve to more accurately approximate selected functions with a smaller amount of sub-intervals.

Assuming a discrete approximation of a continuous distribution function $p(x)$ with n discrete points x_i of subintervals of length $h = (b - a)/n$, one can approximately compute the integral $\int_a^b p(x)dx$ as

$$\int_a^b p(x)dx = h \cdot \sum_{i=0}^{n} p(x_i), \quad x_i = a + i * h\,.$$

In PDBs, several approaches are applied for numerical integration over continuous distribution functions, including histogram-based approximation, representation through characteristic functions, and sampling, as described in Section 2.3.4.

Statistical Measures

Computing the probability that a variable x takes a value in a specified range or exceeds a certain tail quantile t_x is a basic but central functionality for the analysis of risks. As specific cases, one can calculate the upper or lower *tail probability*, i.e., the probability that x exceeds or falls below a threshold value t_x, as the integral $\int_{t_x}^{\infty} p(x)\,dx$ or $\int_{-\infty}^{t_x} p(x)\,dx$, respectively. This means that we compute the value of the CDF at t_x, yielding the result $1 - P(t_x)$ and $P(t_x)$, respectively. Another essential functionality of statistical analyses is the computation of the quantile (percentile) values of a distribution. The quantiles are computed as the value of the ICDF at a given probability value p, $P^{-1}(p)$. The function P^{-1} is defined for continuous and strictly monotonic distributions. The p-quantile, or $(p \cdot 100)^{th}$ percentile, is the value below which $p \cdot 100\%$ of all samples randomly drawn for x will fall.

Inverse Transform Approach Besides general statistical analysis, the ICDF of a distribution can be used for sampling from this distribution: By applying the so-called *inverse transform approach*, one can draw a sample from a distribution $P(x)$ by computing its quantile function $P^{-1}(u)$ at a sample u drawn from a Uniform distribution in $[0,1]$. In PDBs, inversion is relevant foremost for implementing approaches that rely on Monte-Carlo simulations. In this thesis, the inverse transform approach is applied as a central function for processing correlation structures (see Chapter 4).

Aggregation

Similar to historic (deterministic) data, aggregation over probabilistic values is a central functionality, e.g., for the computation of measures over data of finer granularity. For example, given individual forecast values for next year's revenue per product group, a user may want to compute the total prospective revenue of the company as the sum of all individual revenues.

Convolution In the general case, to compute the sum of (two or more) random variables, one *convolves* their distribution functions. The PDF of the sum of random variables is the convolution of their individual PDFs. Likewise, the CDF of the sum of random variables is the convolution of their individual CDFs. Convolution is an integration problem that can be solved efficiently using Fourier transforms. Convolution of integrable functions translates to multiplication in Fourier space, the Fourier transform of the convolution being equal to the product of their individual Fourier transforms:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(f)\,.$$

For example, to compute the CDF of the sum $revenue = rev_1 + rev_2$ of two continuously distributed values rev_1, rev_2, we compute the inverse Fourier transform of the product of the Fourier transforms of the individual CDFs P_{rev_1} and P_{rev_2}. This approach can be applied both to aggregate symbolically represented distribution functions and to their approximate (histogram-based) distribution representations. In the latter case, one again computes a numerical integration.

In the special case of normally distributed values, their convolution can be computed very efficiently. The sum of n normally distributed values $\mathit{Normal}(\mu_i, \sigma_i^2)$ can again be expressed as a Normal distribution. A similar case can be made for Gamma distributed variables, such that

$$\sum_{i=1}^{n} \mathit{Normal}(\mu_i, \sigma_i^2) = \mathit{Normal}\left(\sum_i \mu_i, \sum_i \sigma_i^2\right) \quad \text{and}$$

$$\sum_{i=1}^{n} \mathit{Gamma}(\alpha_i, \beta) = \mathit{Gamma}\left(\sum_{i=1}^{n} \alpha_i, \beta\right)\,.$$

For most distribution functions, however, no such solution is available, or it would be highly expensive to obtain an exact result analytically. In those cases, one can apply numerical convolution over the discrete approximations of the underlying functions, as discussed above.

Approximate Aggregation In cases where a large number of variables must be aggregated, computing their exact sum[2] through convolution can be very expensive. Moreover, a

[2]We denote the computation of convolutions as the *exact* approach, even though the case of numerical convolution involves a certain degree of inaccuracy due to approximation.

user might not even require an exact aggregation result, since approximate results may suffice to draw a conclusion from a query result. Approximate approaches can be used to compute aggregates over large amounts of data based on their moments, such as their *mean* and *variance*. This way, one can, for example, efficiently calculate a low, high, and expected value for aggregation queries such as *COUNT*, *SUM*, *AVG*, *MIN*, and *MAX* (see, e.g., [MIW07]). Similarly, one can approximately calculate confidence interval information for such aggregates, i.e., compute the variance of the resulting aggregate distribution.

A crucial characteristic of those approaches is that they rely completely on the assumption that the underlying values (and therefore, the result) can be appropriately captured through a normal distribution. This, indeed, is often not the case, which must be considered when applying this form of approximate aggregation. In this thesis, we similarly resort to approximate approaches for computing aggregate values, as will be discussed in Section 3.3.3.

Temporal Aggregation To incorporate knowledge about temporal indeterminacy of planning in the analyses, users need methods to aggregate over measure values associated with indeterminate facts (events). For example, in Use Case 1, the user wants to aggregate the product sales prospected for a relevant product within the first half of 2012, while the exact product release date is unknown.

There exist a variety of approaches addressing the problem of *temporal aggregation* of measure attribute values over *certain* time intervals. In this context, the notion of *overlap* from Allen's interval algebra (see Section 3.2.3) is an important basic operation. Overlap is defined as the operation of intersection in time, and can be applied to compute predicates and orderings of temporally indeterminate data (see [All83, DS98]). In particular, if the validity interval associated with an attribute value has zero overlap with an *aggregation interval*, the attribute value will not contribute to the aggregate result. Otherwise, the attribute value will contribute to the aggregation result.

Attributes can have different semantics regarding the relation of their validity interval and the considered aggregation interval. In particular, [BGJ06] refers to *constant* and *malleable* attributes. A *constant attribute* contributes to an aggregate in a constant fashion, i.e., without its value being adjusted to the length of the aggregation interval. An example is the age of an employee; when computing the maximum age of all employees working in a company during some year, their age attribute value is considered constant regardless of the fraction to which their employment *overlaps* the aggregation interval. On the other hand, the value of a so-called *malleable* attribute is adjusted to the overlap fraction between their validity interval and the aggregation interval. An example for a *malleable* attribute is the salary of an employee: When computing the sum of salaries paid to all employees during a fortnight interval, we need to aggregate the fraction of each employee's salary that contributes to the aggregate (i.e., half of their monthly salaries, assuming all employees were employed during the complete fortnight). Considering those semantics, one can apply different aggregations, such as the computation of instantaneous, cumulative, and moving window aggregates over temporal data [BGJ06, SGM93].

In this work, the aggregation over temporally indeterminate data (events) accounts for both uncertain start times and durations. In the context of scenario planning, this is relevant for evaluating different possible developments, so that one can plan activities that have an effect on the start and duration times of the considered events. For example, a user can evaluate the best and worst case $revenue$ for a launched product. Then, if necessary, he can devise activities to prevent late launch dates if they would compromise a minimal revenue target. We address the aspect of aggregation over temporally indeterminate data using both concepts from [DS98] and [BGLS03] in Sections 3.2.3 and 3.3.4.

2.4.2 Data Modifications

To enable users to evaluate assumptions about changing conditions that may influence the result of a scenario analysis, it is necessary to allow for modifications of data as well as the introduction of new uncertain information. For example, when a user prospects a possible future revenue value, he should be able to introduce new information (or modify assumptions) to see the effect of deviations from initial assumptions. New or changed assumptions should thus be reflected through modifications of the initial distributions. Traditional PDBs focus on the evaluation of relational queries or analysis functions, and mostly do not consider modifications to uncertain data. It is important to track applied modifications to data in an analysis process. Therefore, one needs to address modifications in data through a version-based approach, keeping the original data as well as new versions rather than updating the original data in-place. This is pointed out as a central requirement in [SBD$^+$09] (where it is stated for the field of scientific databases). Chapter 5 of this thesis addresses an approach to incorporate modifications to input distributions in the scenario analysis process.

In general, it is important that users can analyze how a final analysis result was computed, keeping track of initial data that served as input to an analysis, as well as information about changes and processing steps applied to this data. This requirement is addressed by the topic of provenance handling, which is addressed in the final section of this chapter.

2.5 Provenance

Conceptually, the concept of *provenance* refers to the capture and management of *bookkeeping* information about how data has been derived by a query or throughout a process. In the preceding sections, the relevance of keeping such provenance information has been pointed out. On the one hand, provenance information is crucial for correct query evaluation under PWS (see Section 2.3.1). On the other hand, one needs provenance to enable users an adequate evaluation and interpretation of scenario data.

The question of managing information about data production, curation, and analysis processes has received broad attention during the last decade. In this section, we give a brief overview of approaches to provenance management, and relate to special aspects particularly relevant in the context of this thesis.

2.5.1 Provenance Management Systems

For a brief overview of different aspects of provenance management, we adopt the categorization of [GD07]. The major categories considered therein relate to the *provenance model*, the *query and manipulation functionality*, and the *storage model*.

The *provenance model* determines the sort of information one can manage. Here, one can distinguish the concept of *source provenance* and *transformation provenance*. Source provenance is information about the data items involved in the derivation of another data item and can be further distinguished, for example, to include all input items used in a derivation, or only those which *positively* contributed to the result (e.g., all tuples that satisfied a predicate). Apart from the original data items, the system may store additional information, such as attached meta data. In each case, the recorded provenance information can be on different levels of granularity, such as the attribute, tuple or relation level. The concept of *transformation provenance* includes all information about the transformations (operations) that have been applied in a derivation process. Again, the captured information can be on different levels of detail and include completely manual, automatic, or semiautomatic transformations as well as additional meta information. As a third aspect of the provenance model, the underlying *world model* is considered. In the closed-world model it is assumed that the provenance management system controls all transformations and data items; in the *open-world model* it may not control all transformations and thus, exact provenance recording may not be possible.

In the context of this thesis, we consider the capture of source provenance and high-level transformation provenance information. We assume a closed-world model, meaning that all performed transformations are known and can be recorded.

The aspect of *manipulation and query functionalities* includes, first, different possibilities to query the abovementioned sorts of provenance data. In the context of scenario planning, a user may, for example, need to pose queries about the base data (historic data as well as assumptions and forecast data) that was involved in the derivation of scenarios, or about analysis steps applied to the data. Conversely, he may be interested to learn about derived scenario data that depends on some base data item, or the different transformations that have been applied to a base data item. Secondly, a system can also provide functionality for manipulating provenance, e.g., for converting different levels of detail of stored provenance data. This aspect is not in the scope of our work.

Finally, as regards the *storage and recording model*, one can make a distinction whether provenance data is kept physically attached (loosely or tightly coupled) to the actual data, or is stored in a dedicated repository. The recording of provenance itself can be controlled by the system or a user, and provenance information can be captured eagerly (i.e., stored with each transformation) or only be provided at query time (but not persistently stored). A final distinction considers the question whether provenance attached to an input data item of a transformation is propagated through the succeeding transformation, and if so, how many such propagations are applied.

In Chapter 5 of this thesis, we describe the capture and use of scenario provenance information. There, provenance information is captured for each operator application and stored in dedicated provenance tables. The major focus of using provenance data in this thesis is on evaluating changes in scenario data, rather than on capturing and querying provenance information as such.

2.5.2 Source and Transformation Provenance

This section addresses selected aspects of provenance both in the field of probabilistic data management, and in the field of workflow management. Since scenario planning includes both data-related and process-related aspects, it is necessary to understand the topic of provenance from both view points.

Provenance in Uncertain Data Management

In the context of PWS, one can consider provenance information as "a constraint that tells us which worlds are possible." [DS07]. As such, provenance information is necessary to ensure correct query evaluation and computation of confidence values of a query result [STW08, ABS$^+$06, SD07, SMS$^+$08] as discussed in Section 2.4. By exploiting provenance information, the confidence computation step can be decoupled from the data computation step, in order to correctly consider dependency information introduced in the process of query processing.

Many systems such as Trio [ABS$^+$06] and Orion [SMM$^+$08] track information about the data derivation on tuple- and attribute-level during query processing. Effectively, they store a dependency set of tuples for every tuple derived in a query. Additionally, predicates applied during query processing are recorded in dedicated tables.

From a slightly different point of view, provenance relates to the representation of dependencies in data, as described in Section 2.3.5. Similar to capturing information about the provenance of a data item, dependency information reflects constraints involving different random variables in the database.

Capturing only the *source provenance* of derived scenario data does not suffice to fully reflect the derivation of data in a scenario analysis process. This is in contrast to many applications of provenance in databases, where the focus is on relational queries, whose processing semantics can be captured adequately using source provenance. As a consequence, current approaches to uncertain data management primarily handle the source provenance of data. In contrast, this thesis considers the handling of *both source and transformation provenance*, similar to approaches taken in the field of *data-centric workflows*.

Data-Centric Workflow Provenance

Workflows are a central technical concept in many areas of research as well as business. A workflow describes a sequence of connected steps (tasks) that describe a process. The individual steps of a workflow can be abstract activities as well as concrete data transformation steps. The *data flow* of a workflow specifically addresses aspects associated with the passing of data between activities. This includes, for example, the specification of data sources, applied transformations (e.g., to clean or merge data), and data consistency constraints. During the last decade, the management of so-called data-centric workflows has become increasingly important for keeping track of the great amounts of information processed and created, e.g., in scientific analysis processes and business processes. Scenario analysis processes can be considered as a form of data-centric workflow, whose provenance needs to be captured and managed appropriately.

In this context, it is necessary to capture the relation between in- and output data of the individual steps in a process, but also information about the transformations (operations) that were applied in the process. In particular, it is necessary to keep track of the invocations of specific analysis operations and their input and output data, in order to enable the user to query the provenance of analysis results and to recompute analysis processes. Different models, such as the Open Provenance Model (OPM), have been devised to enable users to capture the execution of processes adequately. In such models, the provenance of an analysis process is usually being reflected through a graph, where the nodes reflect activities (applied operations) and artifacts (consumed or produced data) and the edges reflect the relations between the different nodes. Based on inference rules, a user can then query the collected provenance of a process, e.g., to discover which data artifacts and transformations were part of the process that led to a final analysis result.

Many of the requirements stemming from workflow provenance management can similarly be applied to the context of scenario analysis processes. Those requirements are:

- The need to determine applied operations and to recompute analysis results based on changes in underlying inputs.
- The possibility to reuse an analysis process or certain sequences therein.
- The need to determine and analyze the source data respectively the derivation of result data in order to learn about the context of analysis results, their quality, or inconsistencies that may arise.
- The possibility to trace the provenance of data for legal or regulatory issues.

Regardless of those common characteristics, research on data-centric workflow management so-far largely addressed the scientific domain. Existing approaches do not address aspects such as the processing of business data residing in a database, the computation of analytic operators over such data, or the recomputation of analysis processes based on changed input data. Those aspects are the central topic of Chapter 5, where the recomputation of analysis processes is addressed using approaches to provenance from both the data management and workflow management fields.

2.6 Summary

In this chapter, different aspects that build the conceptual and technical context of this thesis were considered. The topic of managing performance and risks in enterprises and the specific technique of scenario planning were introduced as the general application field in which the thesis is situated. Then, uncertainty in data was introduced as a crucial issue in planning processes of all sorts, and different relevant aspects of probabilistic data and their management in databases were discussed. This included the representation and processing of uncertain data in difference flavors, and the aspect of dependencies (correlation) in such data. Particularly the latter aspect has been addressed only to a limited degree hitherto, especially with respect to the handling of correlation between *continuous* distributions. To address this gap, the flexible handling of such correlations will play a prominent role in this thesis, and is addressed in detail in Chapter 4. Finally, the topic of provenance was considered and its relevance for scenario planning was pointed out. In Chapter 5 we will exploit both source and transformation provenance in order to enable an efficient recomputation of scenario analyses.

As a basis for the subsequent topics, the next chapter presents the data model used throughout the thesis, and introduces a set of analysis operators which users can apply in an analysis process.

3 Data Representation and Analysis

This chapter describes the data model for representing uncertain data and provides a set of operators for uni- and multivariate analysis. As a starting point, we give a conceptual description of the general scenario analysis process. Then, the representation of uni- and multivariate data, as well as functionality for analyzing and modifying such data in the analysis process, are discussed.

3.1 The Scenario Analysis Process

In the process of scenario analysis users require both operations for OLAP, i.e., slicing, dicing, roll-up, and drill-down, and specific operations for deriving, analyzing, and modifying uncertain data.

A generic *data analysis process* includes activities for accessing and modeling, cleaning and integrating, transforming, and exploring data. The specific steps depend strongly on the specific aim of the analysis. The general aim of any data analysis, however, is to highlight or extract useful information from potentially huge amounts of data, in order to derive conclusions and support decision making. Analogously, the aim of the *scenario analysis process* is to support users in decision making by means of methods to extract and derive potential future business developments (scenarios) based on historic data and hypothesis. As a basic working assumption, we assume that business data are already available and accessible based on multidimensional data model, as noted before in Section 1.3.1.

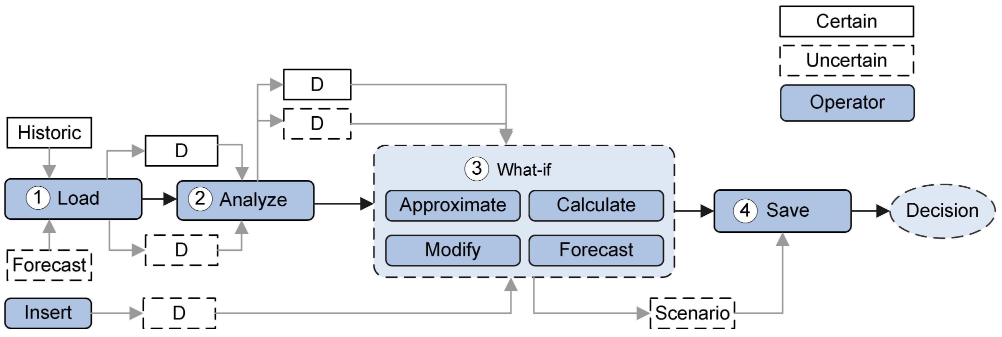


Figure 3.1: A schematic description of the what-if analysis process

Figure 3.1 illustrates on an abstract level how operations are applied in such a process in a sequential and iterative fashion. After loading and analyzing underlying data, the user can apply different operations that represent assumptions regarding possible business developments, before he investigates, compares, and possibly persists the resulting scenarios. In the graph, operators are shown as rectangles with round corners. Rectangles depict data consumed and produced by the steps (i.e., the applied operators) in the process. We consider both certain (historic) and uncertain (approximated or forecast) data loaded from a database, where it is accessible through a dedicated representation (Step 1). In the figure, uncertain data is distinguished from certain data by dashed lines. Users select and analyze data to view it on an appropriate aggregation level (Step 2). To evaluate a possible development of the current situation those views can then be processed through several steps to create an (uncertain) what-if scenario (Step 3). For example, users can select and calculate statistic measures of past data and use the result as input for a forecast. Additionally, they might insert new values assumed to hold for some attribute. They can iterate steps 2 and 3 to derive, explore, and subsequently compare several scenarios based on previous intermediate results. For example, data resulting from an analysis in step 2 can serve as input for several what-if analysis steps 3a, 3b, etc. The final result of the illustrated process is uncertain data of one or several scenarios. Users can investigate those scenarios to make appropriate business decisions and possibly store them for later reference (Step 4).

Analysis Data Flow

Formally, the process of scenario analysis can be viewed as a data flow. In this data flow, operators are applied to one or multiple input data, process the data, and produce output data which may serve as input to further operators or as the result of the data flow.

Data When referring to data artifacts in the context of the analysis process, we use the generic term of a *data item* x. For example, we may refer to a sales forecast value in Use Case 1 as data item x_S. Further, we use X to refer to (ordered) sets of data items $\{x_1, ..., x_n\}$ and use $x_{[i]}$ to refer to the i^{th} item of X. For example, $X_{S[1]}$ would be used to refer to sales the forecast value of the first month.

Since the focus within this thesis is on analysis operations over (numeric) attribute data, x is used to directly refer directly to the value of a (mostly numeric) attribute of an implicitly associated tuple t_i in the database, i.e., $x = t_i.x$. Tuple t_i may naturally have additional values for both measure and dimensional attributes $A = \{a_0, ..., a_n\}$, e.g., storing foreign key values identifying the location or time associated with a forecast. Those attributes can be used for selection, grouping, or joining within the analysis process. It will be implicitly assumed that the association between the data item x and the attribute values of attributes a_j of a tuple t_i are accessible in the given context.

Operators An operator o is applied to (one or multiple) input data sets or items (i.e., sets containing one data item), $o.in = \{X_1, ..., X_k\}$ and produces an output data set or item. For operators o that expect an atomic input item but are applied to an input dataset X, an iterator model similar to [MPB10] is implicitly applied. That is, o is iteratively applied to each item $x_i \in X$ and produces a result item y_i of an output data set $o.out = Y$ in turn. Similar to [ISW10], operators can be composed such that for a composition $o_1 \circ o_2$, the result of operator

o_1 serves as input to o_2, i.e., $Y_2 = o_2(X_2)$, where $X_2 = Y_1 = o_1(o_1.in)$, and similarly for operators that take several inputs.

3.2 Distribution Representation

To appropriately represent uncertainty in decision support systems, one needs to observe the specific characteristics of the data and tasks involved. As this work is situated in the context of scenario analysis processes, both the data representation and processing functionalities are designed support statistical analysis over business data and forecasts. This requires that we handle uncertainty on the level of numeric attribute values rather than, for example, evaluating discrete probabilities associated with a fact in the database. Therefore, the focus is on handling continuously distributed data on attribute-level, rather than discrete alternatives on the level of attribute values or tuples.

Our data representation addresses the following aspects:

- Considering the representation of uncertainty on attribute-level, the domain of a distribution can be either continuous or discrete, and it may be represented through a known distribution family or follow an arbitrary distribution.
- Real-world data are often dependent (correlated), with dependency (correlation) structures that may be of arbitrary form.
- Uncertainty can occur both in measure attributes, i.e., the values that are 'measured' and analyzed, and in dimension attributes, i.e., the values that characterize the context of a measure value, such as its time of occurrence.

3.2.1 Representation Forms

A data item can refer either to an uncertain attribute value or a deterministic value. The case of deterministic values—which may be either fact data in a database table or result values computed throughout the analysis process— will be distinguished through the notation $\dot{x}$ and $\dot{X}$, where it serves clarity.

An uncertain attribute value is represented through a continuous random variable x. The variable x is associated with a distribution of possible values within a support interval $[\underline{x}, \overline{x}]$, where $\underline{x}$ and $\overline{x}$ mark the lower and upper bounds of x, respectively. To enable a broad class of represented distribution forms, both a symbolic and an approximate representation of distributions through histograms are supported.

Symbolic Representation

The first representation form applied is a symbolical representation, which is captured by means of an instance of a distribution family and distribution-specific parameter values for the given distribution. The system presented in this work supports the basic classes of *Gaussian* (normal), *Gamma* and *Uniform* distributions. Users can represent a specific distribution instance for a variable x by specifying its distribution family and parameter values, such as

$x = Gaussian(\mu : 100, \sigma : 30)$ for a variable x that is normally distributed with a mean of 100 and a standard deviation of 30. The use of symbolic representations is beneficial due to their compact representation; however, not every real-world data can be closely fit using a known distribution family. Moreover, only a small number of distribution functions can be evaluated analytically in an efficient manner, or there is a high induced computational effort to do so.

Histogram Representation

To provide a generic representation of distributions, the use of histograms is supported. By using histograms, one can circumvent potential restrictions connected with the symbolic distribution representation regarding (i) their (potentially low) fit to actual data and (ii) to provide an approximate numeric representation of symbolic representations in the course of query processing. When discussing the processing of distribution values in the further course of this chapter, the approximate histogram-based representation form is implicitly assumed. That is, when handling symbolic distributions, the working assumption is that they are either evaluated analytically based on their known distribution function, or their discrete approximation is processed using numeric algorithms.

Histograms may be built both as a frequency distribution (empirical distribution) over sample data, and as an approximation to discrete or continuous distribution functions.

For the first case, for the distribution of x within a support interval $I_x = [\underline{x}, \overline{x}]$ consider a set of pairs (v_j, f_j) where v_j are the discrete sample values from I_x and f_j are the respective frequencies of those values in the underlying data. A histogram $x^{(\beta)}$ is built as an empirical distribution of the underlying data by partitioning the concrete values for x into β mutually disjoint subsets (bins) $B = \{b_1, ..., b_\beta\}$. Each bin b_k represents a sub-interval $[l_k, h_k]$ of I_x and is associated with the frequency f_k computed by aggregating frequencies f_j of all values in $[l_k, h_k]$.[1]

The second way to build a histogram-based representation is by approximating a symbolically represented distribution function of a variable x through a histogram. This functionality may be used explicitly by a user, or can be applied implicitly during the processing of an operator which internally requires a histogram-based representation. Given a variable x with a symbolic distribution function, a corresponding histogram is built by associating each of the β bins $b_k, 1 < k < \beta$ with the density f_k computed as the integral $f_k = \int_{l_k}^{h_k} P(x)dx$ over the density function of x.

This work uses *equi-width histograms*. As stated before, one reason for this choice is their construction and update efficiency. Alternative partitioning schemes such as equi-depth and V-Optimal (see, e.g., the classification in [PHIS96]) on average produce more exact approximations, and could be integrated with most of the approaches presented in this work. However, the construction and updating of the resulting histograms induces larger costs than the use of equi-width histograms. The handling of the arbitrary partition boundaries of, e.g., equi-depth or V-optimal histograms, requires adaptations of the histogram processing routines, which in

[1] Where the concrete nature of the underlying histogram is not relevant or clear from the context, the specification of β will be omitted from the notation and we will simply refer to the variable x.

some cases would further lead to a decreased processing efficiency. Further, the arbitrary partition boundaries prevent some of the optimization techniques proposed in Chapters 4 and 5. An in-depth evaluation of the use of the various histogram classes is out of scope for this work.

Density and (Inverse) Cumulative Distribution functions

The basic functions pdf_x, cdf_x and inv_x are provided to represent random variables x through their PDF, CDF, and ICDF values, respectively. Those basic functions can be accessed explicitly in queries, or used implicitly during the computation of other (more complex) operators. The computation of pdf_x, cdf_x and inv_x occurs transparently over symbolic and histogram-based representations, since we process the former based on their discrete approximation.

Representing Differences (Deltas) between Distributions

For some operations, we want to represent the difference between two distributions (in terms of their distribution), rather than their absolute values. Storing the difference, or *delta*, between two distributions is addressed through a dedicated *delta representation* Δ. In particular, the delta between two data x and x' is denoted as Δ_x, where $\Delta_x(v)$ is the difference of the cumulative distribution functions cdf_x and $cdf_{x'}$ at value v. For simplicity, we will write Δ_x for $\Delta_{x,x'}$ between an original item x and a new version of the same item, x', in case the context is clear.
Note that for any two variables x, x' the value of $\Delta_x(v)$ is always in $[-1, 1]$ for any value of v.

Delta Histograms As a particular representation of Δ, we use so-called *delta histograms* Δ^H. Given x and x', their delta information Δ_x^H is computed as a histogram that comprises, for each modified bin, the bin-wise differences of $x^{(\beta)}$ and $x'^{(\beta)}$, i.e., $f_{\Delta_x,j} = f_{x',j} - f_{x,j}$. Conceptually, the resulting representation Δ_x^H reflects a stepwise function that represents the difference between the cumulative distribution functions of x and x' in the modified interval. Therefore, strictly speaking Δ_x^H is neither a histogram, nor a distribution, since its value is *negative* where $cdf_{x'}(v) < cdf_{x'}(v)$. Still, we refer to Δ_x^H as a *histogram* in the context of this thesis. Indeed, we can process the delta representations in a way that incorporates their integration with analysis results computed over standard histogram representations. In particular, we will exploit delta representations for an efficient recomputation of analysis results under changes in input data, as discussed in Chapter 5.

3.2.2 Multivariate Distributions

In addition to uncertainty associated with individual attributes, real-world data is often subject to correlation in data. This means that values can not be handled as independent from each other as they covary and therefore exhibit a dependent behavior which must be considered in computations. Examples of analysis functions over multidimensional distributions are the computation of joint tail probabilities and marginal distributions, which are highly important in our context due to the need to analyze potential positive and negative effects between different key figures.

Here, we first consider the general representation of multidimensional data, while specific approaches to handling the underlying correlation will be discussed at a later point.

Let a multidimensional distribution over m random variables be characterized through

(a) its m marginal distributions, which we will denote as $x.dim_j$ and

(b) the correlation structure that defines the dependencies between the marginal distributions, referred to as $x.cor$.

Empirical Multivariate Distributions Similar to the univariate case, multivariate distribution can be derived from values of m (possibly correlated) attributes in a database. The result is an m-dimensional histogram, which again serves as a generic representation of arbitrary distribution forms, analogously to the one-dimensional case.

Symbolic specification of multivariate distributions In cases where a user wants to freely define a joint distribution, or the amount of data required to derive an empirical distribution is not available (or no correlated data at all is available), we allow the user to provide a specification of both $x.dim_j$ and $x.cor$. That is, the correlation component can be defined depending on a desired correlation. For example, the user could specify that a revenue value x_r and a marketing value x_m both follow normal distributions with means μ_r, μ_m and that their correlation is defined by a covariance matrix Σ. Then, the resulting joint distribution can be represented symbolically as $x_{r,m} = Gaussian(\mu : [\mu_r, \mu_m], \Sigma)$, where $x_{r,m}.dim_1 =$ Gaussian$(\mu_r, \Sigma(1,1))$,$x_{r,m}.dim_2 =$ Gaussian$(\mu_m, \Sigma(2,2))$, and $x_{r,m}.cor = \Sigma$.

Handling generic correlation structures The basic way to specify the correlation information $x.cor$ is to use a covariance matrix. This approach, however, requires that (a) the correlated values are normally distributed, and (b) the correlation between the marginals is *linear*. Those are restrictions that do not apply for many real-world data. We therefore consider the topic of multivariate distributions in a more generic manner in Chapter 4, where we address the representation, construction, and processing of joint distributions by means of a self-contained representation of their underlying correlation structures.

3.2.3 Indeterminate Dimension Data

Besides uncertainty in the data which is typically the primary target of analysis (i.e., the attributes commonly addressed by the term *key figures* in the OLAP context) there may also exist uncertainty in the *dimensional attributes* of the data under analysis. In the context of forecasting and planning, the temporal dimension naturally is highly relevant: Users may neither know the exact start time nor the duration of considered events (product launches, deliveries, etc). Still, we want to enable users to analyze data (such as the costs of deliveries) associated with such events along time.

Temporally Indeterminate Events By the term of a *temporally indeterminate event* $e = (t, d, x)$ we refer to a *fact* in the database which represents some event in the real world, and whose start time t and duration d can not be determined with certainty, and thus are indeterminate. Further, an event is associated with a key figure x, which reflects a quantity of interest associated with e, e.g., a cost induced by a service event or revenue generated through a sales event.

Analogous to the representation of random variables that describe measure values, we represent indeterminacy by means of indeterminate *instants* t and *durations* d, similar to [DS98]. Both are defined over the domain of time; the smallest time unit representable (in a specific application context) is called a *chronon*. For a random start time variable t^{start}, we specify its support interval $I_{t^{start}}$ of possible chronon values and a corresponding distribution over $I_{t^{start}}$ (again, in symbolic or histogram-based form). Accordingly, an indeterminate duration d is associated with a support interval I_d and a corresponding distribution over a possible duration.

Each possible allocation of an event e to a specific start $t_p^{start} = v_p, v_p \in I_t$ and duration $d_q = v_q, v_q \in I_d$ then yields a possible occurrence interval $I_{pq} = [v_p, v_p + v_q]$. Since t and d are assumed to be independent, the probability of e_i occurring exactly during the interval I_{pq} computes as $p(t_p, d_q) = p(t_p) \cdot p(d_q)$. The complete distribution of t^{end} then results as the *convolution* (see Section 2.4.1) of t^{start} and d.

EXAMPLE: Figure 3.2 depicts histograms for an uncertain start time (date) t^{start}, duration d, and the resulting end time distribution t^{end} of a single event e. The event has start times distributed between 10 and 13 and a duration between 2 and 4 days. Given the distributions, the probability for e occurring exactly in $I_{2,2} = [12, 12 + 4]$ is, for example, $p(t_2, d_2) = p(t_2) \cdot p(d_2) = 0.5 \cdot 0.5 = 0.25$.

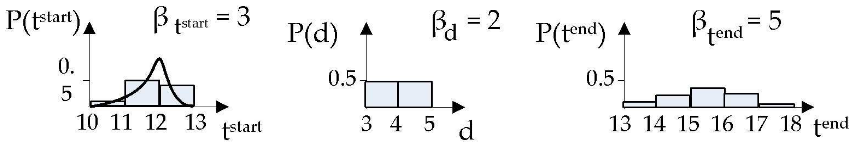


Figure 3.2: Indeterminate start, duration, and end time of e

The representation of uncertain data has now been discussed in its different aspects. The next Section addresses functionality for the analysis of such data by means of different operators. There, we will also consider analytic functionality that takes into consideration the indeterminate temporal association of events.

3.3 Operations

In this section, the provided functionality for the creation, analysis, and modification of uncertain data is discussed. This includes basic operators for

- computing statistical measures of distributions underlying random variables

- deriving, converting, and modifying distributions
- approximate aggregation, and
- the analysis of multivariate distributions.

3.3.1 Statistical Measures

In the following, we consider the set of operations for analysis, introduction, and modification of uncertain values represented through random variables x. As stated before, the operators process histogram-based representations $x^{(\beta)}$ or discrete approximations of symbolically represented distributions, respectively.

Moments

The basic operators $\mathbb{E}(x)$ and Var(x) serve to compute the first two moments, i.e., the mean and variance, of the distribution of x based on its PDF and CDF. We compute the mean (expected value)

$$\mu = \mathbb{E}(x) = \int_{-\infty}^{\infty} x \cdot p(x)\, dx$$

numerically over a histogram representation (assuming uniform spread in each bin) as

$$\mathbb{E}(x) = \sum_{k=1,\ldots,\beta} m_k \cdot p_k, \quad with\, m_k = (l_k + h_k)/2$$

Likewise, we compute the variance

$$\sigma^2 = \mathbb{E}[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 \cdot p(x)\, dx$$

numerically as

$$\text{Var}(x) = \sum_{k=1,\ldots,\beta} (m_k - \mu)^2 \cdot p_k, \quad with\, m_k = (l_k + h_k)/2\,.$$

Complexity: The computation of both $\mathbb{E}$ and Var is done in one pass over all β bins of $x^{(\beta)}$, thus implying costs linear in the granularity of the histogram representation, i.e.,

$$\mathbb{E}(x^{(\beta)}) \;\in\; \mathcal{O}(\beta)$$

$$\text{Var}(x^{(\beta)}) \in \mathcal{O}(\beta)$$

Range and Tail Probability

The operator $\mathcal{T}$ computes the probability of a variable x satisfying predicate $\omega(\upsilon_1, \upsilon_2)$, which is defined as $x > \upsilon_1 \wedge x < \upsilon_2$, i.e., evaluates to true if x is within the range $[\upsilon_1, \upsilon_2]$ of its domain. $\mathcal{T}(x, \omega)$ is computed as the integral $\int_{\upsilon_1}^{\upsilon_2} p(x)dx$, i.e., $\mathcal{T}(x, \omega) = cdf_x(\upsilon_2) - cdf_x(\upsilon_1)$. For the corresponding histogram representation $x^{(\beta)}$, we integrate numerically over the affected region of I_x, i.e., we compute

$$\mathcal{T}(x, \omega(\upsilon_1, \upsilon_2)) = \sum_{k=1,...,\beta} f_k \cdot p_k, \quad \text{with} f_k = f^{\cap}([l_k, h_k], [\upsilon_1, \upsilon_2])$$

where $f^{\cap}$ defines the fraction of $[l_k, h_k]$ that lies in $[\upsilon_1, \upsilon_2]$, i.e., $f_k = 1$ for all bins completely contained in $[\upsilon_1, \upsilon_2]$, and 0 for all bins where $h_k < \upsilon_1 \vee l_k > \upsilon_2$.

If υ_1 is set to $-\infty$ or υ_2 is set to ∞, we compute the lower or upper tail probability of x (denoted as predicates $\omega_<(\upsilon_2)$ and $\omega_>(\upsilon_1)$, respectively). This results in $\mathcal{T}(x, \omega_<) = cdf_x(\upsilon_2)$ and $\mathcal{T}(x, \omega_>) = 1 - cdf_x(\upsilon_1)$, respectively. Evaluating tail probabilities is a central element of risk analysis, where the tail probability is used to reflect the likelihood that an extreme event will occur.

Complexity: The imposed complexity for computing the probability of x satisfying ω is determined by a maximum of 2 calculations of cdf_x (for the lower bound υ_1 and upper bound υ_2). This results in complexity

$$\mathcal{T}(x^{(\beta)}, \omega) \;\in\; \mathcal{O}(\log_2 \beta)$$

3.3.2 Derivation and Conversion

Rather than analyzing existing uncertain information, users may also introduce uncertainty based on underlying deterministic fact data. The operator $\mathcal{D}(F, tgt)$ serves the derivation of a distribution over (historic) values of fact data, which then can either be investigated visually or serve as input for further analyses. The assumption is that a distribution derived from historic facts may constitute a proper reference for the development of another value in some similar context. For example, in our use case UC 1 (Section 1.2), the analyst wants to utilize knowledge about the past revenue 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} as a reference distribution.

The operator $\mathcal{D}(F, tgt)$ receives a number of $n_F = |F|$ of facts F from a fact table of the database. The type of target distribution that shall be derived is specified via the tgt parameter, including the representation type and further parameters. 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, optionally, the lower and upper support (l, h) of the desired histogram $x^{(\beta)}$. In the latter case, he must provide the assumed family of the target distribution. Note that this requires some insight or expectation about the nature of the underlying fact data.

Derivation of Histogram Representations When deriving a histogram over the fact values $F = \{f_1, ..., f_{n_F}\}$, we assume they are provided in a non-sorted order. To construct an equi-width histogram, bin boundaries are statically computed based on a desired lower and upper support (l, h) and the number of bins β and assign each of the n_F values, resulting in complexity $O(n_F)$. If l and h are not specified, they must first be retrieved as the minimum and maximum of all fact values in $\mathcal{O}(n_F)$. [2]

Complexity: The complexity for deriving an equi-width histogram over n_F samples is

$$\mathcal{D}(F, hist(\beta)) \;\in\; \mathcal{O}(n_F)$$

EXAMPLE: Referring to Use Case 1, the user derives a distribution of the assumed sales revenue per store in region R_{new} from facts of the reference region R_{ref}.

In the general case (i.e., if bin boundaries can not be computed statically), 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.

Complexity: Given arbitrary histogram bin boundaries, the complexity for deriving a histogram over n_F samples is

$$\mathcal{D}(F, hist(\beta)) \;\in\; \mathcal{O}(n_F \cdot log_2(\beta))$$

Derivation of Symbolic Representations If users know about the nature of fact data, they may derive a symbolic distribution representation as an instance of a specific distribution family. Uniform, Gaussian, and Gamma distributions are supported as instances of common distribution families. Further distribution families may be added if required. Finding the lower and upper bounds of the support interval of an assumed uniform distribution requires determining their maximum and minimum in $\mathcal{O}(n_F)$. For a normal distribution, we similarly need to process each of the n_F values to incrementally compute the mean and variance. For a Gamma distribution,we estimate the scale and shape parameters from the sample mean and standard deviation using the method of moments estimation.

Complexity: For all considered distribution functions, the computation is implemented in one or multiple runs over the fact data, and thus, implies a complexity of

$$\mathcal{D}(F, symb) \;\in\; \mathcal{O}(n_F)$$

Conversion of Representations As stated above, symbolic representations are converted implicitly to their discrete approximation for further processing in (more complex) operators. In addition, users may apply an operator $\mathcal{DC}(x, tgt)$ to explicitly convert distribution representations. Another application of $\mathcal{DC}(x, hist(\beta))$ arises when the user wants to test a (derived) symbolic distribution representation for goodness of fit with actual data. This is because such tests (e.g., the χ^2-test) mostly rely on binned data, and distributions which shall be tested must therefore be converted to a suitable histogram representation.

[2] Alternative to a user-specified parameter, an optimal β can be estimated at the cost of an additional run over facts F using a basic heuristic aiming at an optimal approximation of the interval with β bins (e.g., Sturges' rule)

EXAMPLE: Referring to use case 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 normal distribution with appropriate mean and variance. In order to further process this value, it may be internally converted to a histogram, depending on the operator to which it is provided as input.

Similar to $\mathcal{D}$, 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 To construct a histogram $x_h^{(\beta)}$ as $x_h^{(\beta)} = \mathcal{DC}(x, (hist, \beta, [l, h]))$, the user specifies the desired number of bins β and the lower and upper support (l, h) of the target histogram. If no lower and upper support are provided, they are retrieved as the value of a low and high quantile value of the source distribution, that is, they are computed as $l = inv_x(q_l)$ and $h = inv_x(q_h)$. The values q_l and q_h for determining the minimum and maximum quantiles can be set parametrically; by default, they are set to $q_l = 0.001$ and $q_h = 0.999$, i.e., the distributions are cut off at their 0.001 and 0.999 quantiles. The very extreme tails are in this case not reflected in the resulting histograms. Based on l, h, and β, the bin boundaries l_k and h_k are then computed and the source distribution is integrated within the lower and upper bin boundary for each of the β target bins b_k to yield the mass lying within b_k, i.e.,

$$p_k = \int_{l_k}^{h_k} pdf_x dx$$

Complexity: The complexity of converting a distribution to a histogram is

$$\mathcal{DC}(x, hist(\beta)) \;\in\; \mathcal{O}(\beta)$$

where the integration costs may vary with the type of the converted distribution function.

Histogram into Symbolic Distribution When converting a histogram to a uniform distribution, the support interval of the Uniform is determined simply from the lower and upper bounds of the histogram, inducing constant access costs in $O(1)$. To convert the histogram to a suitably parameterized normal distribution or Gamma distribution, we need to access boundary and frequency values for all bins b_k to compute the required parameters (mean and variance for a normal distribution or scale and shape for a Gamma distribution, respectively).

Complexity: The conversion to a symbolic representation induces a complexity of

$$\mathcal{DC}(x^{(\beta)}, symb) \;\in\; \mathcal{O}(\beta)$$

Histogram into Histogram As a third alternative, one can change the granularity of discrete approximations, computing $x_2^{(\beta_2)} = \mathcal{DC}(x_1^{(\beta_1)}, hist(\beta_2))$. This conversion serves, e.g., to provide users with a changed granularity of information or is internally applied to convert histograms to the same bin resolution for further processing, such as testing the fit of two distributions to each other. The conversion proceeds by finding the bin boundaries of the β_1 bins of $x_1^{(\beta_1)}$ and computing the area covered by the β_2 bins in $x_2^{(\beta_2)}$ within the boundaries of each bin of $x_1^{(\beta_1)}$.

Complexity: The imposed complexity of the computation is determined by the source and target resolution, thus

$$\mathcal{DC}(x^{(\beta_1)}, hist(\beta 2)) \;\in\; \mathcal{O}(\beta_1 + \beta_2)$$

3.3.3 Aggregation

The operators $\mathcal{A}_{SUM}$ and $\mathcal{A}_{MIN/MAX}$ serve the computation of SUM, MIN, and MAX aggregates over a dataset X. All three aggregations are computed in an approximate fashion. In the case of SUM, we use a general approach also taken, e.g., in [KK10, KLD11], and add up the *expected values* of each $x_i \in X$. That is, we compute a deterministic expected aggregate value $\dot{x}_{SUM}$ as

$$\dot{x}_{SUM} = \mathcal{A}_{SUM}(X) = \sum(\mathbb{E}(x_i)) \quad x_i \in X$$

Complexity: Summing the expected values of all $x_i \in X$ results in a complexity of

$$\mathcal{A}_{SUM}(X = \{x_i^{(\beta)}\}) \;\in\; \mathcal{O}(|X| \cdot \beta)$$

Alternatively, computing exact sums over all distributions $x_i \in X$ by applying fast discrete convolution algorithms (e.g., using Fast Fourier Transform, see Section 2.4.1) results in costs of $\mathcal{A}_{SUM}(X = \{x_i^{(\beta)}\}) \in \mathcal{O}(|X| \cdot \beta \log \beta)$.

The operators $\mathcal{A}_{MIN}$ and $\mathcal{A}_{MAX}$ similarly to $\mathcal{A}_{SUM}$ use an approximate approach, yielding a deterministic output value as minimum or maximum of a set X, respectively. The results of $\mathcal{A}_{MIN}$ and $\mathcal{A}_{MAX}$ are computed based on low and high quantile values $inv_{x_i}(q_l)$ and $inv_{x_i}(q_h)$ of the distributions associated with each $x_i \in X$. Again, the values of q_l and q_h can be provided by the user; by default, they are set to $q_l = 0.01$ and $q_h = 0.99$. Those values are based on the reasoning that extreme outliers shall not be considered in the computation of the maxima. In cases where this reasoning is not suitable, the parameters can be adjusted accordingly.

Then, we compute the minimum and maximum of all values $x_i \in X$ as

$$\dot{x}_{MIN} = \mathcal{A}_{MIN}(X) = \min(\{inv(0.01, x_i) | x_i \in X\})$$

and

$$\dot{x}_{MAX} = \mathcal{A}_{MAX}(X) = \max(\{inv(0.99, x_i) | x_i \in X\}),$$

respectively.

Complexity: The imposed complexity of the computation depends on the costs of inv_{x_i} for each x_i, thus resulting in a complexity of

$$\mathcal{A}_{MIN/MAX}(X = \{x_i^{(\beta)}\}) \in \mathcal{O}(|X| \cdot \log_2 \beta)$$

3.3.4 Indeterminate Temporal Aggregation

In order to enable the handling of temporal indeterminacy of plans, we consider indeterminate events as described in Section 3.2.3. We use the operator $\mathcal{A}^{\underline{T}}(E, \underline{T})$ to compute the aggregate (sum, minimum, or maximum) of a considered key figure associated with temporally indeterminate events within a time interval $\underline{T} = \left[l_{\underline{T}}, h_{\underline{T}}\right]$. The set $E = \{e_i\}$ contains indeterminate events $e_i = (x_{kf,i}, t_i, d_i)$, each defined through an associated key figure $x_{kf,i}$, and variables t_i, d_i which represent uncertain start and duration times of e_i, respectively. To compute an 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 step for a single event $e \in E$ (omitting the subscript for reasons of readability).

Summation $\mathcal{A}^{\underline{\mathbf{T}}}_{\mathbf{SUM}}$: We first focus on the computation of $\mathcal{A}^{\underline{T}}_{SUM}$. We address only the case where x_{kf} has so-called interval-aggregate semantics (which is called a "malleable" attribute in [BGJ06]). This means, the value of x_{kf} is considered as a sum over the complete duration d of the associated event e. Therefore, if only a fraction of the event duration lies within $\underline{T}$, the same fraction of x_{kf} is considered as contribution in the summation procedure of $\mathcal{A}^{\underline{T}}_{SUM}$.

Following this reasoning, we need to compute the overlap fraction $f^{\cap}$ as the relative part of *potential occurrence intervals* of e that overlap $\underline{T}$. Then, the contribution of e to the aggregate result amounts to $f^{\cap} \cdot x$. The fraction $f^{\cap}$ depends both on the position of $\underline{T}$ and on the set of all possible intervals $I_{pq} \in \mathcal{I} = \left\{\left[t^{start}_{pq} = v_p, t^{end}_{pq} = v_p + v_q\right]\right\}$, $v_p \in I_t, v_q \in I_d$ in which e can occur. We compute all possible occurrence interval overlaps with $\underline{T}$ for event $e \in E$ as follows:

$$overlap_{pq} = \begin{cases} (h-l)/d_q & \text{if } t^s_p \leq l \wedge t^e_{pq} \geq h \\ (t^e_{pq} - l)/d_q & \text{if } t^s_p \leq l \wedge t^e_{pq} > l \\ 1 & \text{if } t^s_p > l \wedge t^e_{pq} < h \\ (h - t^s_p)/d_q & \text{if } t^s_p < h \wedge t^e_{pq} > h \\ 0 & \text{else} \end{cases}$$

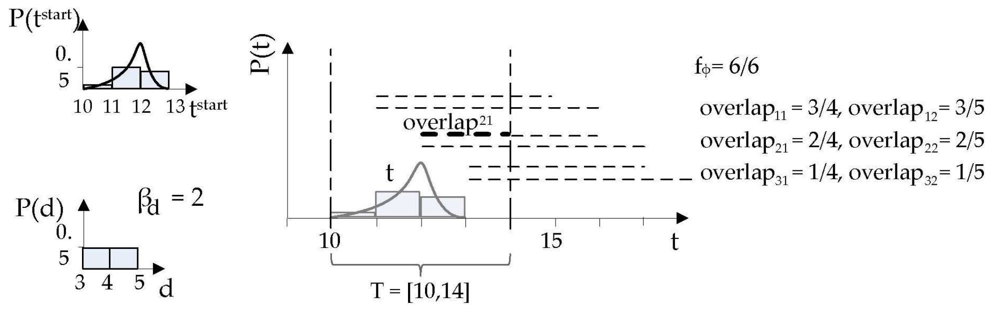
To compute $f^{\cap}$, we also require the occurrence probabilities of all possible occurrence intervals. Since we consider t and d independent, the occurrence probability $P(t = v_p, d = v_q)$ is given by the product $P(t = v_p) \cdot P(d = v_q)$.

In the aggregation procedure, we apply this computation for a relevant set of events $e_i \in E$ and then compute contributions $f_i^{\cap}$ of e_i as

$$f_i^{\cap} = \sum_{p,q} overlap_{ipq} \cdot P(t_i = v_p, d_i = v_q)$$

and yield the final result as $\mathcal{A}^{\underline{T}}_{SUM}(E, \underline{T}) = \sum_i \mathbb{E}(x_{kf_i}) \cdot f^{\cap}_i$.

Example: Figure 3.3(a) depicts, for the single event e, the start time t, $\beta_t = 3$ and duration d, $\beta_d = 2$ of e while Figure 3.3(b) shows how aggregation works over this single event. For event e, $f^{\cap}$ 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 each I_{pq} that lies within $\underline{T} = [1, 5]$.



(a) Indeterminate start and duration

(b) Aggregation over indeterminate events for event e

Figure 3.3: Representation and processing of temporal indeterminacy associated with indeterminate events

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 overlap $\underline{T}$ (and therefore contribute to the aggregate result) as f^{ϕ}.

Complexity: Computing the sum over indeterminate events E, with an average number of potential occurrence intervals n_I and an average aggregation interval overlap of f^{ϕ}, implies costs for computations of overlaps and joint probabilities in the complexity

$$\mathcal{A}^{\underline{T}}_{SUM}(E, \underline{T}) \in \mathcal{O}(|E| \cdot n_I \cdot f^{\phi})$$

Extrema $\mathcal{A}^{\underline{\mathbf{T}}}_{\mathbf{MIN/MAX}}$: For computing $\mathcal{A}^{\underline{T}}_{MIN}$ and $\mathcal{A}^{\underline{T}}_{MAX}$, we consider the set of contributing events $E^{\cap} \in E$ to be exactly those events $e_i \in E$ that have *any* potential occurrence interval I_{ipq} that overlaps $\underline{T}$. This is true for all e_i where $l_{t_i} < h_{\underline{T}} \wedge h_{t_i} + h_{d_i} > l_{\underline{T}}$. Now, we use $f^{\cap}$ to refer to the fraction $f^{\cap} = |E^{\cap}|/|E|$. We then compute the resulting aggregate over all $e_i \in E^{\cap}$ similar to the standard minimum or maximum computation discussed in Section 3.3.3 over the associated values x_{kf_i}. That is, for the set $X^{\cap}_{kf} = \{x_{kf_i} | e_i \in E^{\cap}\}$ we compute the minimum as

$$\mathcal{A}^{\underline{T}}_{MIN}(X_{kf}, \underline{T}) = \mathcal{A}_{MIN}(X^{\cap}_{kf})$$

and the maximum as

$$\mathcal{A}^{\underline{T}}_{MAX}(X_{kf}, \underline{T}) = \mathcal{A}_{MAX}(X^{\cap}_{kf})$$

Complexity: Computing the minimum and maximum of measure values $x^{(\beta)}_{kf_i}$ associated with events $e_i \in E$ within a time interval $\underline{T}$ incurs constant costs of $\mathcal{O}(1)$ for determining whether an event belongs to $E^{\cap}$, followed by the computation of $\mathcal{A}_{MIN/MAX}$ over all those $e_i \in E^{\cap}$:

$$\mathcal{A}^{\underline{T}}_{MIN/MAX}(E, \underline{T}) \;\in\; \mathcal{O}(|E| \cdot f^{\cap} log\beta)$$

3.3.5 Modification of Distributions

Modifying distributions is necessary to introduce new assumptions about the concrete distribution of a random variable in a scenario or adapt its distribution if the old distribution deviates from actual evidence.

For example, if the user assumes the tails of a distribution irrelevant for his current analysis (or does not want to consider this part of the distribution in his scenario), he can update the value by modifying (parts of) the distribution. In this case, the modification can be represented symbolically, i.e., by *flooring* [SMS$^+$08] the distribution in its tail. Here, we focus on the modification of histogram-based representations of distributions. That is, we change the densities associated with relevant bins of the histogram associated with a variable x. In the case of flooring, the densities of the bins corresponding to the tail region of x are conceptually set to 0. Likewise, we allow to change individual parts of a distribution to reflect lower or higher densities in the corresponding part of I_x.

We provide an operator $\mathcal{U}(x, x', \omega)$ to update a distribution x into a modified distribution x' within an interval determined by the predicate ω. An application of $\mathcal{U}$ causes the frequencies associated with affected sub-intervals of I_x to be changed to the corresponding density in x', optionally restricted to the interval specified by ω. This way, a user can explicitly specify both the affected bins and their target density through x'. If no information is given for x', the predicate ω specifies the distribution part to be "floored". If no information is given for ω, the complete support interval of x will be affected by the modification.

To ensure that modified values can be traced back to the original value, we must not replace x by x'. Rather, we keep the original value together with a reference to the *delta information* Δ_x represented by means of a *delta histogram* Δ^H_x, as introduced in Section 3.2.1. In the worst case, i.e., when the modification affects the complete interval of x, we need to compute and store β bin density values for Δ^H_x.

Complexity: The costs for computing and storing modifications depend on the resulting degree of modification (f_m), i.e., the fraction to which a distribution, represented by $x^{(\beta)}$, is actually affected by the condition ω.

$$\mathcal{U}(x^{(\beta)}, f_m) \;\in\; \mathcal{O}(f_m \cdot \beta)$$

The topic of modifications in input data is a central issue in the context of provenance handling and recomputation of scenarios over changed input data. In this context, the representation of modifications by means of deltas can be exploited for an efficient change impact analysis over modified input data. In particular, using Δ_x^H, we can compute the *deviation* between original and the modified data, x and x', at any position v, as $\Delta_x^H(v)$ in constant time, similar to the computation of cdf_x. In Section 5, we will come back to this topic and address the question how modifications can be efficiently incorporated in the recomputation of queries over modified data or under assumed deviations to assumptions.

3.3.6 Multivariate Analysis

In this section, we briefly discuss the application of operators over multidimensional distributions, which serve to derive relevant statistic measures over jointly distributed variables. Such analyses are particularly relevant in risk analysis, where users want to consider risks and chances associated with joint developments of portfolio assets, product sales, etc. The handling of correlation information will be addressed in depth in Chapter 4, where we also discuss possibilities for an efficient processing of the subsequent operations under arbitrary correlation structures.

At this point, recall from Section 3.2.2 that we consider a joint distribution x_{jnt} to be specified through its m marginal distributions $x_{jnt}.dim_j, j = 1, ..., m$ and an m-dimensional correlation structure $x_{jnt}.cor$. For now, we assume $pdf_{x_{jnt}}$, $cdf_{x_{jnt}}$ are implemented based on a corresponding multidimensional histogram $x_{jnt}^{(\beta)}$, where each dimension of $x_{jnt}^{(\beta)}$ reflects the domain of the marginal distribution corresponding to that dimension.

Joint Tail Probabilities

We use the operator $\mathcal{T}^m(x_{jnt}, \Omega)$ to compute the probability of jointly occurring events, such that each $\omega_j \in \Omega$ is satisfied. Each $\omega_j(v_{j,1}, v_{j,2}) \in \Omega$ is a predicate on dimension $x_{jnt}.dim_j$, as described for the univariate case in Section 3.3.1. Specifically, we compute the upper joint tail probability of x_{jnt} when using predicates $\omega_{j_>}(v_{j,1})$ and setting all $v_{j,2}$ to ∞, and the lower joint tail probability when using predicates $\omega_{j_<}(v_{j,2})$ and setting all $v_{j,1}$ to $-\infty$. The results of $\mathcal{T}^m$ are then the joint probability values

$$p_{v_j}^{>} = \mathcal{T}^m(x_{jnt}, \Omega = \{\omega_>(v_j)\}) = \int_{v_1}^{\infty} \dots \int_{v_m}^{\infty} pdf_{x_{jnt}}(x_1, \dots x_m)\, dx_1 \dots dx_m \quad and$$

$$p_{v_j}^{<} = \mathcal{T}^m(x_{jnt}, \Omega = \{\omega_<(v_j)\}) = \int_{-\infty}^{v_1} \dots \int_{-\infty}^{v_m} pdf_{x_{jnt}}(x_1, x_2)\, dx_1 \dots dx_m$$

for the upper and lower joint tail probability, respectively.

Example: In the risk analysis context, considering the joint probability of events is a crucial functionality. Jointly occurring high losses can pose an existential risk, e.g., to an insurer or in stock trading, and must be considered appropriately. In use case Use Case 2, the analyst wishes to compute the risk that two claim categories, flood claims x_f and hurricane claims x_h, jointly *exceed* a claim amount of $v_f = \$15bn$ and $v_h = \$15bn$, respectively. Given their joint distribution is represented through $x_{f,h}$, we compute $p^u_{\upsilon_f,\upsilon_h} = \mathcal{T}^m(x_{f,h}, (\omega_f, \omega_h))$ using $\omega_{f_>}(\upsilon_f), \omega_{h>}(\upsilon_h)$.

Complexity: The imposed complexity of joint probability computations, assuming x_{jnt} is accessed through a multidimensional histogram $x^{(\beta)}_{jnt}$, where each of the m dimension being represented through β bins, is exponential in m:

$$\mathcal{T}^m(x^{(\beta)}_{jnt}, \Omega) \;\in\; \mathcal{O}(\beta^m)$$

Marginal Distributions

The operator $\mathcal{M}(x_{jnt}, \Omega)$ serves to compute a marginal distribution x_{jnt} from a joint distribution x_{jnt} given predicates $\omega_j(\upsilon_{j,1}, \upsilon_{j,2}) \in \Omega, 1 \leq j < m$ on j dimensions $x_{jnt}.dim_j$ of x_{jnt}. The result is an $(m-j)$-dimensional distribution with all dimensions dim_j *marginalized* (or: projected) out.

In the scenario and risk analysis context, the user can apply $\mathcal{M}$ to compute the conditional marginal distributions for variables of interest under different assumptions about the correlated variables.

EXAMPLE: For example, for analyzing the joint distribution $x_{f,h}$ of potential insurance claims from Use Case 2, the user can compute $x_{f'} = x_{(f|h>\$7\,bn)}$ as the marginal distribution of x_f under the condition that x_h will take on a very high value, e.g., exceed $\$7\,bn$. The result is a one-dimensional (conditional) distribution $x_{f'}$. Another example is the computation of conditional distribution of sales revenues by applying different assumptions regarding economic growth or marketing expenditures, as addressed in Use Case 1.

Complexity: The imposed complexity of computing marginal distributions, assuming x_{jnt} is accessed through an m-dimensional histogram $x^{(\beta)}_{jnt}$, each of the m dimension being represented through β bins, is

$$\mathcal{M}(x^{(\beta)}_{jnt}, \Omega) \;\in\; \mathcal{O}(\beta^m)$$

3.4 Summary

This chapter first discussed the nature of the scenario analysis process and the requirements for data representation and processing therein. For a representation of distributions associated with random variables in an analysis, we allow both for a symbolic and a histogram-based representation. For its generic nature, the histogram-based form can be used to model (approximate) arbitrary distributions, including the approximation of symbolically represented distributions. A set of relevant basic operations for computations over uni- and multivariate distributions was then discussed in the context of the scenario analysis process. Those include the derivation and conversion of distributions from source fact data, "standard" aggregation as well as the temporal aggregation of values associated with temporally indeterminate events, and the computation of statistical measures such as expected values, quantiles, and tail probabilities. Finally, we turned to the analysis of multivariate distributions, for which operators for computing joint tail probabilities and conditional marginal distributions were discussed.

Analyzing joint distributions, such as between the sales revenues or stock prices of different assets in a portfolio, is highly relevant for risk analysis purposes as part of the scenario analysis. Therefore, the issue of modeling and processing the correlation structures underlying data is addressed in more depth in the following chapter.

4 Correlation Handling

An important point in the process of analyzing risks and opportunities in a business is the correlation underlying many real-world aspects. Investigating the effect of correlation on business developments is an important means to help experts to prevent misestimations of risks and opportunities.

Section 3.3.6 already discussed the computation of joint probabilities and marginal distributions over multidimensional distributions, under the assumption that those distributions are readily available. However, correlation information is not always present in the data a user wants to analyze. At times, only independently represented distributions are available, e.g., in the case of forecasts that have been produced separately for two measure values. The user still may want to analyze such data under the assumption of a correlation structure that is present in *historic* data. In this case, one must allow him to first extract the correlation structure (which may be of arbitrary form) in order to subsequently introduce it to the data of interest. In other cases, it is not possible to compute any correlation structure since there is no historic data, or the given data is too sparse (e.g., data about the occurrence of extreme catastrophe damages as indicated in Use Case 2 in Section 1.2). In this case we want to support the introduction and querying of *assumed* correlation structures in order to enable the user to investigate their potential effects over the correlated data.

This chapter focuses on a number of specific requirements for correlation handling in the risk analysis context, and introduces a generic approach to representing and processing correlation.

4.1 Requirements for Correlation Processing

Recall from Section 3.3.6 that representing correlation in data through the covariance information (i.e., the covariance matrix one can compute based on available correlated samples) between the variables relies on a number of assumptions. First, for the results to be valid, the correlated variables must be (nearly) normally distributed, i.e., one cannot represent joint distributions between arbitrary distributions. Second, the dependency between the marginal distributions is assumed to be linear. This means that one cannot analyze correlations that are non-linear, which often occur in real-world data. Both assumptions imply that the basic covariance-based approach has a very restricted applicability for *representing* correlations. In the same way, one cannot apply this approach to *introduce* correlation information of *arbitrary* form between *arbitrary* correlations. Therefore, a more generic approach to correlation handling is required.

To clarify specific requirements for handling correlation and to illustrate the techniques presented in this chapter, let us first focus in more detail on different aspects of Task B of

Use Case 2, first presented in Section 1.2. The general topic of this task is the analysis of data reflecting claims for different kinds of insurance damages. The goal is to evaluate the claim data in a way that takes information about correlation between them into consideration. That is, the *joint occurrence* of claims from different claim classes is considered. We now consider two alternative cases for Task B. In the first case, if no historic reference information is available, but the user still wants to introduce correlation information as an assumption in his analysis. In the second case, sufficient historic claim data is available, and the user can extract information about their correlation from the data. Subsequently, he wants to apply this correlation to hitherto uncorrelated data (such as forecasts).

Task B.1 The first case concerns damages caused by large disaster events, namely, hurricanes and floods. For such events, the available data is usually sparse due to the low case numbers. Figure 4.1 shows losses associated with flood events (f) and hurricane events (h) in the US, recorded for a period of twenty years[1]. For lack of detailed data, a constant loss of $\$1\,bn$ is being associated with each hurricane event. It is further assumed that the distributions of flood and hurricane damages, x_f and x_h, are Gamma distributed and their parameters are fit according to the available data.

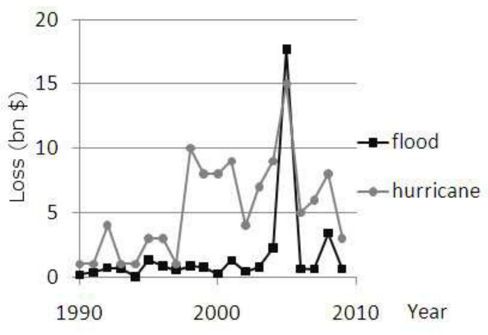


Figure 4.1: Disaster-related claims

From Figure 4.1, an analyst can see that the data correlate (as one may expect from the natural characteristics of such catastrophe events) and, using the available data points, he can compute a linear correlation coefficient of 0.76. However, due to the sparsity of the underlying data he cannot determine the actual correlation *structure* nor use such information for analyses such as the computation of marginals or joint probabilities.

For example, the user may wish to compute the expected value of x_f under the assumption that x_h will take a high value (e.g., $x_h > \$7\,bn$), or the risk that both flood and hurricane events cause damages above a certain threshold. Since no correlation structure can be faithfully extracted from the data, the user should be able to consider alternative settings, using different potential correlation structures as assumptions. Then, he could investigate the re-

[1]http://www.fema.gov/business/nfip/statistics/pcstat.shtm

sulting joint distributions either visually or through further analysis, e.g., by calculating the probability (risk) that $x_f > \$15\,bn \land x_h > \$15\,bn$.

Task B.2 Now, we consider a class of insurance for which much higher case numbers are usually reported. Particularly, assume the user evaluates insurance losses caused by burglary and vehicle theft (whose distributions are denoted as x_b and x_v). He wants to learn about the correlation structure of the historic insurance data in order to then incorporate this historic correlation in the analysis of forecast values, represented by distributions x_{bf} and x_{vf}. As example data, we consider yearly crime numbers in the USA[2] over a period of 20 years (1990-2009) as indicators for the number of claims put forward to an insurance. For lack of fine-grained data[3] the distribution of claims throughout the year is assumed to be uniform, and weekly claim numbers are simulated accordingly. Figure 4.2 displays their joint distribution $x_{b,v}$, which clearly indicates some correlation pattern in the data. Now, for a *joint* analysis of x_{bf} and x_{vf}, the user wants to take the historic correlation pattern into account. To this end, he needs to be enabled to *extract* the correlation information. Further, the user might want to investigate *different* correlation structures, such as to learn about changing correlation patterns throughout time, or in different regions or customer groups of interest.

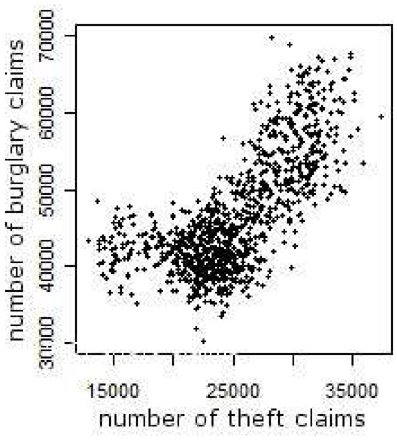


Figure 4.2: Crime-related claims

Both analysis tasks necessitate the flexible introduction and analysis of correlation information between arbitrary distributions—each of x_f, x_h, x_b, x_v, x_{fh} and x_{bv} could follow any distribution—and its representation in a form that can be processed efficiently. The first task specifically requires means to extract correlation patterns from historic data, and to make them available for further processing, while the second task focuses on the handling of precomputed

[2]http://www2.fbi.gov/ucr/cius2009/data/table_01.html

[3]Note that analyses often involve much more fine-grained data, e.g., on a per-client basis, thus processing potentially large volumes of historic data. This aspect will be addressed in the evaluation.

structures stored in the database. To serve those requirements, the remainder of this chapter presents approaches for a user to

- Represent arbitrary correlation structures over arbitrarily distributed data,
- Extract information about arbitrary correlation structures flexibly from historic data, enable their storage and reuse,
- Apply an arbitrary correlation structure to arbitrary continuous distributions to yield a joint distribution that exhibits the introduced correlation structure, and
- Query the produced joint distributions efficiently.

Most previous work on correlation handling in databases addresses basic forms of tuple dependency or dependencies between discrete attribute value alternatives. Examples for such correlations are implication (i.e., perfectly positive correlation), linear correlation, or mutual exclusion between discrete values. Handling (producing and analyzing) joint distributions with arbitrary (particularly, non-linear) correlation structures over arbitrary (particularly, non-normally) distributed continuous variables has not been treated in depth in the database domain so far.

We borrow the approach of using so-called copulas, which is a well-established statistical technique applied, e.g., in the field of finance and insurance. The use of copulas is supported partially by tools for statistical analysis, such as MatLab, Mathematica and R (see, e.g., [Li06, KY10]) or can be implemented using statistical functions. In the next sections, we discuss their characteristics and, particularly, their merits regarding the previously described requirements for correlation handling. Section 4.2 gives the theoretic basis of the copula concept. Then, we will consider the central concepts for representing and processing copula-based correlation structures at the database level in Sections 4.3 and 4.4.

4.2 Copulas

A copula (Latin for "link") essentially represents the relation between individual distributions and their joint distribution (for a detailed introduction, see, e.g., [Nel06]).

4.2.1 Definition and Characteristics

The basic idea of the concept of copula is that a specific m-dimensional joint distribution can be represented as a joint distribution over the uniform transforms of its m marginal distributions, i.e., in the interval $[0, 1]^m$. Recall from Section 3.2.2 the definition of a multivariate distribution x through its dimension components $x.dim_j$ and a correlation component, $x.cor$. In brief, in the context of this thesis the copula encodes the correlation component $x.cor$, while each component $x.dim_j$ can be specified to be a distribution of *arbitrary* form. The copula is in itself an m-dimensional distribution function $C : [0, 1]^m \to [0, 1]$ that represents an m-dimensional correlation structure on the unit cube $[0, 1]^m$, and thus independently from any specific marginals.

In the following, without restriction of generality, copulas are discussed in a bivariate setting, unless stated otherwise. That is, we consider copulas with uniform marginals u and v for linking $m = 2$ univariate distributions x_1 and x_2.

Formally, copulas are founded on Sklar's theorem [Skl59, MST07], which can be stated as follows for the bivariate case:

Theorem 1 (Sklar's theorem) *Given H as a bivariate distribution with $F(x) = H(x, \infty)$ and $G(y) = H(\infty, y)$ as univariate marginal distributions, there exists a (copula) function $C : [0,1]^2 \rightarrow [0,1]$ so that $H(x,y) = C(F(x), G(y))$. That is, there exists a copula C relating F and G so as to form a joint distribution H with F and G as marginals.*

In brief, the most important benefit of copulas are

- their independence from the specific (marginal) distributions to correlate,
- their ability to model any correlation (both regarding its structure and degree), and
- their invariance to linear transformations.

Using Sklar's theorem and exploiting the named characteristics, one can build a copula that represents the dependency of any bivariate distribution, and vice versa, apply this copula to "couple" any two distributions with the represented correlation. In terms of the definition of a multivariate distribution representation, this means that we can *map* the dependency modeled through C in $[0,1]^m$ to a *specific* joint distribution x_{jnt} between m marginals through the functions $inv_j : x_C.dim_j \rightarrow x_{jnt}.dim_j$. Conversely, we can use cdf_j to map the distribution function from x_{jnt} to $[0,1]^m$ through $cdf_j : x_{jnt}.dim_j \rightarrow x_C.dim_j$. This methodology offers a flexible means both to represent existing correlation patterns (extracted from historic data) and to introduce such patterns to uncorrelated data.

Below, the construction of copulas is discussed, before we address their application,i.e., the introduction of correlation structures between univariate data.

4.2.2 Copula Construction

Copulas can be constructed in two ways depending on the available information. If no historic data is available, one can generate a copula from a specific copula family. If, however, sufficient amounts of historic data are available, one can empirically estimate a copula that reflects their correlation. Below, we discuss the copula construction from a known copula type and the derivation of empirical copulas in more detail. The first approach can serve to apply pure *assumptions* about a possible correlation to the existing data; the second approach is especially valuable to extract *specific* correlation structures from historic data in order to enable their evaluation in the context of other data. For example, in the first task of the insurance use case, a user can derive multiple empirical copulas over time windows of the historic claim data in order to evaluate changes in their dependency structure.

Parametric Construction

By constructing copulas from a standard copula family, one can represent an assumption about both the *structure* and *strength* of a correlation. For example, if the user has no detailed information about the correlation between the values of x_f and x_h, as is the case in Task B.1, it can be a helpful alternative for him to investigate different assumptions about possible correlations. In the following discussion, we use copulas from the families of elliptical and Archimedian copulas. In the case of elliptical copulas, the copula is derived from (samples of) a known multivariate distribution function (such as a bivariate Gaussian or Student-T-distribution) exploiting Sklar's theorem. One yields a bivariate copula C as $C(u, v) = H(F^{-1}(u), G^{-1}(v))$. For example, to construct a Gaussian copula, a bivariate distribution H of two standard normal variates F and G with a desired correlation coefficient can be used. Samples taken of H are then transformed to $[0, 1]$ through the inverse of the standard normals, i.e., F^{-1} and G^{-1}. Alternatively, if the user wants to represent a correlation structure which is especially strong in the tails of the marginals, he can use a T-copula, constructed from a bivariate T-distribution, instead. In the case of Archimedian copulas (such as a Clayton copula), one directly applies a closed-form generator function to build the copula, instead of sampling from a distribution.

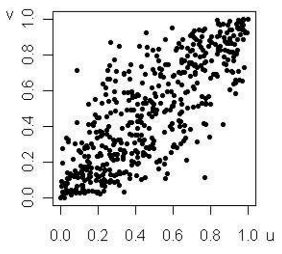
In the following, the term $C_{H,d}$ denotes a copula built using a function H with a given correlation factor d.

EXAMPLE: Figure 4.3(a) shows a Gaussian copula $C_{G,0.8}$ with a correlation factor of 0.8, while Figure 4.3(b) shows a copula $C_{T(1),0.8}$ based on a T-distribution with one degree of freedom and the same correlation factor. Note how the structure of correlation differs—$C_{T(1),0.8}$ shows a higher dependency in the tails of the marginals (i.e., a larger number of samples in the upper right and lower left corners of $C_{T(1),0.8}$), despite the same factor of correlation in both cases.

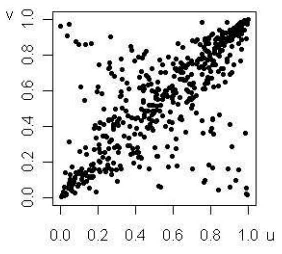
Deriving Copulas from Fact Data

When sufficient observations are available for related measures, a user can first extract knowledge about their correlation in the form of a copula in order to compare different structures or transfer this knowledge to other data in a second step. For example, in Task B.2, the user is given $|F|$ correlated samples for weekly burglary and vehicle theft claims (b_i, v_i, $i = 0, ..., |F|$). Suppose now that the user wants to apply their correlation structure to results of separate forecasts, say, x_{bFc} and x_{vFc}.

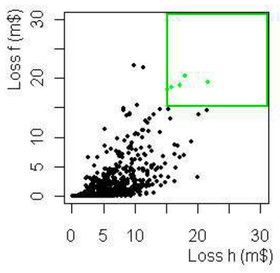
To derive a copula from the historic data, either a (semi-) parametric approach can be applied to fit a copula of a given type, or an empirical copula can be derived from the fact data. The primary benefit of empirical copulas is that the functional form of the correlation structure represented in the result copula is derived *directly* from the data. Therefore, it is not necessary for the user to introduce any further assumptions about the shape of the correlation. To build an empirical copula, one replaces, for each of the $|F|$ m-dimensional samples (here, $m = 2$), the coordinate values (here, (b_i, v_i)) by their *ranks* divided by $|F|$; conceptually, this corre-



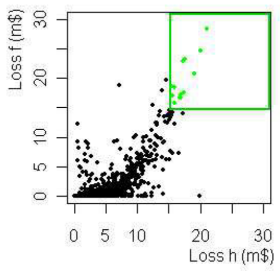
(a) $C_{G,0.8}$



(b) $C_{T(1),0.8}$



(c) $x_{f,h}$ using $C_{G,0.8}$



(d) $x_{f,h}$ using $C_{T(1),0.8}$

Figure 4.3: Copulas (using 1000 samples) representing different structures of correlation and corresponding correlation results

sponds to transforming each of the m marginal sample distributions to $[0, 1]$, resulting in a discrete distribution in $[0, 1]^m$. This way, one can capture any correlation pattern, including specific tail dependencies, asymmetries, etc., in the data. In contrast to the empirical approach, the use of (semi-)parametric approaches implies that one first makes an assumption about the appropriate copula type (e.g., a Gaussian copula). Then, as the name suggests, the parameters for that particular type are fit to the data.

In this work, only the empirical derivation of copulas is applied, since it is the most generic approach and does not require any further information from the user.

4.2.3 Applying Correlation

To correlate two distributions x_1 and x_2 based on a copula C, we again exploit Theorem 1, now substituting F and G with x_1 and x_2. That is, we invert the samples (u_i, v_i) previously derived for copula C by computing the inverse distribution function values of x_1 and x_2. This results in samples $(inv_1(u_i), inv_2(v_i))$ of the result joint distribution $D_{1,2}$. Figures 4.3(c) and 4.3(d) show the joint distribution $x_{f,h}$ that results from correlating the distributions x_f and x_h by using a copula $C_{G,0.8}$ and $C_{T(1),0.8}$, respectively. One can observe the higher tail dependency in Figure 4.3(d) as opposed to Figure 4.3(c) analogous to the applied copulas in Figures 4.3(b) and 4.3(a). For illustration, the right upper corners of $x_{f,h}$ have been marked, corresponding to the occurrence of jointly occurring high claims ($x_f > 15\,bn \wedge x_h > 15\,bn$).

The generic and flexible construction and application of both empirical and parametric copulas emphasizes the benefit that lies in their self-contained dependency representation. Their applicability to arbitrary distributions distinguishes copulas from approaches that represent dependencies specific to the correlated values .

The next sections investigate how the described concepts for constructing and applying copulas serve us to extend the basic representation of multivariate distributions at the database, enabling users to

- **Extract** correlation structures from historical data, as well as accessing parametric correlation structures (Section 4.3),
- **Introduce** correlation structures cor between distributions $x_j, j = 1, ..., m$, resulting in a joint distribution x_{jnt}, with $x_{jnt}.dim_j = x_j$ and $x_{jnt}.cor = cor$ (Section 4.4), and
- **Analyze** the resulting joint distributions or extracted correlation structures.

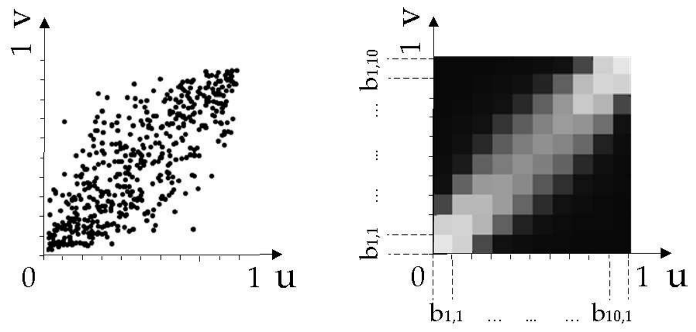
As stated in Section 3.3.6, apart from serving users for visual exploration of distribution characteristics, a joint distribution x_{jnt} can be further processed through the operators $\mathcal{T}^m$ for computing a joint probability over x_{jnt}, as well as the operator $\mathcal{M}$ for computing a marginal distribution over x_{jnt}. For example, to address Task B.1 of the discussed insurance use case, the user could compute $x_{(f|h>7)}$ as the marginal distribution of x_f under the condition that x_h will take on a high value, i.e., exceed $\$7\,bn$. Generally speaking, both $\mathcal{T}^m$ and $\mathcal{M}$ can serve users to create and subsequently compare different possible scenarios over multivariate data. The combination with the flexible introduction of correlation structures gives users even more possibilities to investigate the data under different correlations.

4.3 Approximate Correlation Representations

To apply the copula concept at the database, one could directly use the techniques described in 4.2.2 and 4.2.3 to sample and successively apply a copula. This approach requires that different statistical functions for sampling the desired copulas are available at runtime, and can be executed efficiently. This restriction may often not apply. The alternative approach taken in this work is to use an Approximate Correlation Representation (ACR).

An ACR is a self-contained artifact in the form of an m-dimensional (equi-width) histogram built over the samples S that are drawn or generated for the underlying copula. ACRs serve us to approximate and store correlation structures based on copulas at the database, and access them efficiently at query time. The granularity of the ACR is determined by the number of bins α per dimension, totaling a number of α^m bins for the ACR. For example, for a 2-dimensional copula C the corresponding ACR $\overline{C}$ has α^2 bins $b_{l,k}$, each of which is associated with a density value $w_{l,k}$ that reflects the fraction of all copula samples (s_1, s_2) that fall in this bin of $\overline{C}$.

For an example ACR, see Figure 4.4(a), which shows the ACR $\overline{C}^{10}_{G,0.8}$, with a granularity of $\alpha = 10$ in $m = 2$ dimensions. In the figure, the two-dimensional histogram of $\overline{C}^{10}_{G,0.8}$ is displayed as a heatmap, with lighter shades of gray representing higher density values.



(a)

Figure 4.4: A copula $C^{10}_{G,0.8}$ (1000 samples) and a corresponding ACR histogram $\overline{C}^{10}_{G,0.8}$ with 10^2 bins $b_{l,k}, b, k = 1, ..., 10$

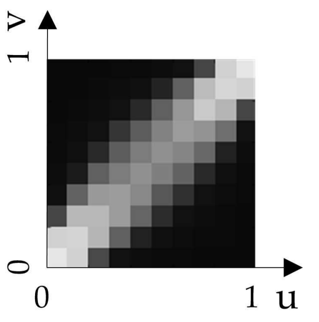
Similar to the parametric and empirical construction of copulas, the construction of ACRs can also be performed in two ways. First, an ACR can be built from a parametrically constructed copula and stored at the database for later access. Second, one can empirically derive a copula which is then approximated by an ACR and can be accessed in further computation steps at query time, similar to the parametric case. Both approaches are discussed in detail below.

4.3.1 Using Precomputed ACRs

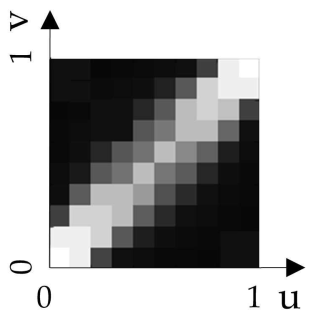
The parametric construction of copulas involves potentially costly multivariate sampling functions. To be independent from external statistical functions and to avoid the sampling costs at runtime, the copula information can be made directly available by means of precomputed structures. To this end, the correlation structures are precomputed externally — i.e., outside the database — and provided as independent artifacts to be stored and accessed at query time in the database. For each desired correlation structure, we first draw a high number of samples (e.g., $n_S = 100k$) from a copula (e.g., a Gaussian copula) to get an accurate representation of the chosen copula. In a second step, an ACR is built over those samples. The accuracy of the produced ACR will be determined from the total number of samples drawn and the parameter α. For example, for an ACR representing a 2-dimensional copula, one could derive a 2-dimensional histogram with 20^2 bins from a total of $100k$ copula samples. Following this approach, one can precompute a number of ACRs for different structures and degrees of correlation, which are then stored and indexed in the database where they can be accessed at query time. Analogous to the notation for copulas, we write $\overline{C}^{\alpha}_{H,d}$ to denote the ACR of a copula $C_{H,d}$ with $\alpha \cdot \alpha$ bins.

EXAMPLE: Figures 4.5(a) and 4.5(b) show the ACRs $\overline{C}^{10}_{G,0.8}$ and $\overline{C}^{10}_{T(1),0.8}$ representing the copulas from Figures 4.3(a) and 4.3(b), respectively.

The benefit of using ACRs as a precomputed structure is twofold. Firstly, as the copulas do



(a) $\overline{C}^{10}_{G,0.8}$



(b) $\overline{C}^{10}_{T(1),0.8}$

Figure 4.5: Example ACR histograms (lighter shades of gray represent higher density values)

not have to be built at query time, one achieves significantly shorter processing times, and secondly, one becomes independent from statistical libraries and distribution-specific functions at runtime. For low-dimensional copulas, the storage of their corresponding ACRs incurs only relatively small memory costs due to their small size. For example, the storage of a bidimensional ACR using 20 bins per dimension incurs storage costs for a total of 400 integer and double values, which hold the information about the bins and their associated density.

Complexity: The costs for constructing an m-dimensional ACR are induced by (i) drawing n_{ACR} samples for the desired copula, (ii) building a histogram with α^m bins, and (iii) storing the histogram. As the bin boundaries are statically determined at a constant distance of $1/\alpha$, the complexity for sampling, histogram construction, and storage resolves to

$$ACR(n_{ACR}, m, \alpha) \in \mathcal{O}(\alpha^m + m \cdot n_{ACR})$$

As stated, the costs for drawing or generating an individual sample naturally vary depending on the desired copula and the applied statistics library.

4.3.2 Extracting Empirical ACRs

As a second option, an ACR can be constructed based on an empirical copula derived from historic data (which in itself can already be considered an approximate representation of the correlation structure). The derivation of empirical copulas takes place at runtime based on selected fact data from which the correlation structure shall be extracted.

The operator $\mathcal{E}$ supports this functionality. Note that we again focus on the case of 2-dimensional data, while the presented approaches can be easily extended to the m-dimensional case where $m > 2$. Using $\mathcal{E}(F, \alpha)$ we extract an empirical copula $\overline{C}^{\alpha}_{F}$ with α^2 bins from historical correlated fact values $F = \{(f_{1,i}, f_{2,i})\}$, $i = 0, ..., N$, such as the data representing claims for vehicle theft and burglary in Task B.2. To build the empirical copula, $\mathcal{E}$ first computes the marginal distributions (i.e., x_1 and x_2) and then transforms the samples $(f_{1,i}, f_{2,i})$ to the unit square $[0,1]^2$ by computing the uniform transforms, i.e., $(cdf_{x_1}(f_{1,i}), cdf_{x_2}(f_{2,i}))$.

The construction of an ACR over the empirically derived copula is similar to the case of parametrically derived copulas. By storing empirically derived copulas, users can reference the correlation structure in further queries, or compare different empirical correlation structures. For example, one can extract multiple copulas from data in different time windows to discover temporally evolving correlation structures or, likewise, for data associated with different regions, customer groups, etc. Thus, the user is enabled to evaluate correlation structures over historical data in a typical OLAP setting.

Complexity: The cost for extracting an m-dimensional ACR with α bins per dimension from facts F depends on three steps: First, m empirical marginal distributions $x_j^{(\alpha)}$ are derived, where the bounds of each marginal are determined as the minimum and maximum of all $f_{j,i}$, respectively. The histograms are built by sorting each $f_{j,i}$ in the corresponding bin of $x_j^{(\alpha)}$ based on static bin bounds. As the second step, each sample $(f_{1,i}, ..., f_{m,i})$ is transformed into a copula sample $(u_{1,i}, ..., u_{m,i}) = (cdf_{x_1}(f_{1,i}), ..., cdf_{x_m}(f_{m,i}))$. As a third step, each such sample is sorted into the corresponding target ACR bin, which causes constant cost for each of the m dimensions. Altogether, the complexity of this process is

$$\mathcal{E}(F, m, \alpha) \quad \in \quad \mathcal{O}(m \cdot |F| \cdot \log_2 \alpha)$$

We will show in Section 4.5 how the extraction of empirical copulas can be parallelized, thus making $\mathcal{E}$ a more efficient operation even in the context of large underlying data sets.

4.3.3 Restrictions Imposed by the ACR Approach

Due to the histogram-based storage, the required storage size and access times increase exponentially with the number of dimensions of the underlying copula. This restricts the application of ACRs to multivariate data of relatively low dimensionality ($m < 5$) in practice. Within the scope of the presented work, it can be argued that this low dimensionality is sufficient for many kinds of analysis. In particular, it suffices for enabling users to get ad-hoc insight into complex correlation structures over continuous data over a few variables. This covers a large space of applications, since often, the user's focus is on a small number of key figures. For queries over higher-dimensional data, which may be required for more complex evaluations, e.g., in the field of portfolio analysis, the application of the sampling approach described in Section 4.2 may be more appropriate. Yet, the sampling-based approach comes at the cost of higher runtime requirements, as evaluated in Section 7.

Another potential drawback lies in the fact that ACRs store accumulated density values within discrete bin boundaries, i.e., in an approximate manner. Thus, their application inadvertently introduces some inaccuracy. As stated in [JXW$^+$08], analyses of probabilistic data inherently involve uncertainty. Therefore, it can be argued that a certain amount of inaccuracy in such analyses is acceptable. The accuracy of query results for applying different degrees of approximation for an ACR is evaluated in Section 7.5.2. The evaluation shows that, the approximation error of ACRs indeed has only a small effect on the query results in most cases.

4.3.4 Using Nested ACR Histograms

To increase the accuracy of an ACR, one can approximate different copula regions at varying levels of granularity. A nested (hierarchical) partitioning scheme is used to achieve this aim. That is, selected bins b within the overall ACR histogram are being represented through a *nested histogram* of finer granularity. This way, one can represent such copula regions that are of particular interest in the analysis very accurately. For example one can represent the *joint tails* of the dependency structure with increased accuracy. Figure 4.6 illustrates this approach via a simple example, where a copula is represented through an ACR with $\alpha = 5$, and its joint tail bins ($b_{1,1}$ and $b_{5,5}$) are represented through one nested histogram with $\alpha_{1,1} = 10$ and $\alpha_{5,5} = 2$ bins per dimension, respectively. Representing the complete ACR in the same accuracy as $b_{1,1}$ would require a total of 100^2 bins. The use of different granularities per bin thus enables a more accurate representation of relevant regions while retaining relatively low memory requirements for storing the respective ACRs or keeping them in memory. In the example, a total of $10^2+5^2+2^2 = 129$ bins are provided for an overall highly accurate copula approximation, with finer-grained approximation being provided in the marked tail regions.

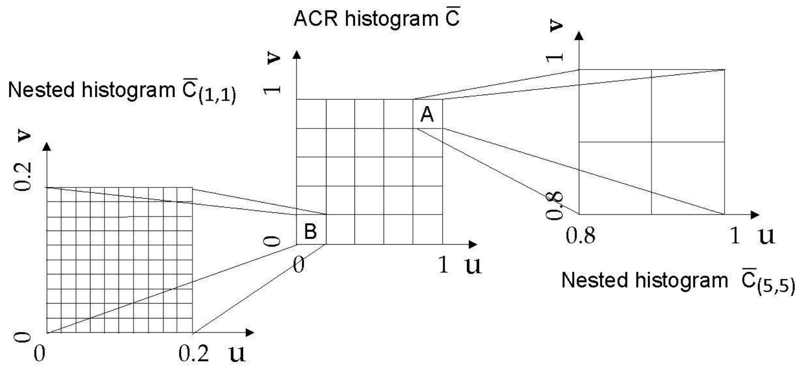

ACR histogram $\bar{C}$
Nested histogram $\bar{C}_{(1,1)}$
Nested histogram $\bar{C}_{(5,5)}$
A
B
v
u
0
1
0.2
0.8

Figure 4.6: Nested histogram representation

Using this hierarchical histogram approach approach, highly relevant parts of the copula, e.g., reflecting the joint tail densities, can be represented and processed with increased accuracy, as will be shown in Section 7. It should be noted that alternative partitioning schemes could of course be used for this purpose. This includes more complex partitioning schemes that result in arbitrary partitions such as equi-depth or V-Optimal [MPS99]. Such schemes could, on the one hand, yield an even more accurate representation of the complete copula. On the other hand, as stated in [Sir05], the resulting histograms induce larger costs than the use of equi-width histograms. This concerns the construction and updating of ACR histograms as well as storing and accessing the resulting histogram representations. In particular, while arbitrary partitions require that the boundaries of all bins are stored and accessed at query time,

a hierarchical equi-width partitioning requires only to store the frequency values for each bin, (plus potential references to nested histograms).
Another aspect that supports the choice of nested equi-width histograms is that a user may desire to approximate *selected* copula regions highly accurately when precomputing ACRs, rather than having the partitioning determined by a generic heuristic.

Having discussed different forms of (pre-)computing ACRs, the following section discusses the introduction of the represented correlation structures in data based on a given ACR. Note that the presented approach works for any kind of ACR representation, irrespective of whether a basic ACR histogram or a nested histogram representation are used, or whether a completely different partitioning is applied during the ACR construction.

4.4 Introducing Correlation

Each ACR—either parametrically or empirically derived—serves as a self-contained artifact that encapsulates a potential correlation structure and serves as input when correlation shall be *introduced* between two or more distributions. For example, to introduce a correlation with degree 0.8 and a heavy dependency in the tails of the marginals between two distributions x_1 and x_2, the ACR $\overline{C}^{\alpha}_{T(1),0.8}$ can be selected from the set of precomputed ACRs and applied for correlation introduction.
The term *correlation introduction* means that a joint distribution x_{jnt} is produced between distributions $x_j, j = 1, ..., m$ using the provided correlation structure $\overline{C}$ such that for the resulting joint distribution, we have $x_{jnt}.dim_j = x_j$ and $x_{jnt}.cor = \overline{C}$.

The operator $\mathcal{C}(x_{j=1,...,m}, \overline{C})$ takes as input the m distributions $x_{j=1,...,m}$ and the correlation structure $\overline{C}$ that shall be introduced to yield x_{jnt}. This is achieved essentially through the method described in Section 4.2.3. However, instead of inverting individually derived samples, $\mathcal{C}$ uses the aggregated sample information represented by each bin of the applied ACR. In the following, two methods for processing the aggregated information, implemented by operators $\mathcal{C}^{basic}$ and $\mathcal{C}^{resample}$, are described. Further, an even more efficient processing scheme, implemented by operator $\mathcal{C}^{reverse}$, is provided. For reasons of readability, the approaches will be described for the $(m = 2)$-dimensional case again, i.e., for producing a joint distribution $x_{1,2}$ by correlating x_1 and x_2. The complexity for each approach will be discussed for the generic case, i.e., for arbitrary m.

4.4.1 Basic Approach

The basic approach for correlation introduction is implemented by the operator $\mathcal{C}^{basic}$ as outlined in Listing 4.1. There, the assumption is that each ACR bin $b_{i,j} \in \overline{C}$ represents one sample to be inverted, and each such sample is weighted according to the bin frequency value, $w_{i,j}$. Consider the application of the ACR $\overline{C}^{10}_{T(1),0.8}$ to two distributions x_1, x_2 to yield a result distribution histogram $x_{1,2}^{(\beta_1,\beta_2)}$, where $\beta_1 = \beta_2 = 10$. We choose the center of each bin $b_{i,j}$, $i, j = 1, \ldots, 10$ in $\overline{C}^{10}_{T(1),0.8}$ as the coordinates (u_i, v_j) of its representative sample and set $w_{i,j}$ to its associated density value in $\overline{C}^{10}_{T(1),0.8}$. We then invert the center coordinates, yielding $x_{1,i} = inv_{x_1}(u_i)$ and $x_{2,j} = inv_{x_2}(v_j)$ as the quantile values of x_1 and x_2 at u_i and v_j, respectively. Finally, we increase the density of the bin associated with $(x_{1,i}, x_{2,j})$ in $\overline{D}_{x,y}$ by $w_{i,j}$.

Listing 4.1: Basic ACR processing through $\mathcal{C}^{basic}$

```
$\mathcal{C}^{basic}(x_1, x_2, \overline{C})$:
  for each $b_{i,j}$ in $\overline{C}$
    $u_i$ = $center_u(b_{i,j})$, $v_j$ = $center_v(b_{i,j})$
    $(x_{1,i}, x_{2,j})$ = $(inv_{x_1}(u_i), inv_{x_2}(v_j))$
    add sample $(x_{1,i}, x_{2,j})$ with weight $w_{i,j}$ to $x_{1,2}$
```

To compare the costs induced by the ACR-based correlation introduction, we first consider the costs of applying the "native" sampling-based approach (referred to as $\mathcal{C}^{sample}$) that samples a copula C at runtime and applies it for correlating m distributions $x_j^{(\beta)}$.

Complexity: First, we need to draw (or generate) n_S m-dimensional samples of the underlying bivariate distribution (or generation function) in $\mathcal{O}(n_S)$. We then transform each such sample using the cumulative distribution functions cdf_{x_m} of the m marginals, summing up to $m \cdot n_S$ calls to statistical functions. Applying the constructed copula implies n_S quantile computations for the desired marginal distributions $x_j^{(\beta)}$ in m dimensions ($\mathcal{O}(n_S \cdot m log_2 \beta)$), totaling to a complexity of

$$\mathcal{C}^{sample}(x_1^{(\beta)}, ..., x_m^{(\beta)}, C, n_S) \ \in \ \mathcal{O}(m \cdot n_S log_2 \beta)$$

In contrast, for the basic ACR-based operator $\mathcal{C}^{basic}$, the costs are reduced as follows:

Complexity: As the procedure of $\mathcal{C}^{basic}$ saves the copula construction costs at runtime and needs to compute only the quantile values for α bin center coordinates for each of the m desired marginal distributions $x_j^{(\beta)}$, the complexity is:

$$\mathcal{C}^{basic}(x_1^{(\beta)}, ..., x_m^{(\beta)}, \overline{C}^{\alpha}) \ \in \ \mathcal{O}(m \cdot \alpha \log_2 \beta)$$

Under the assumption $m \cdot \alpha \ll m \cdot n_S$, this implies lower costs when using $\mathcal{C}^{basic}$ as opposed to $\mathcal{C}^{sample}$, as indicated by the results presented in Section 7. The costs for the histogram construction for the approach of $\mathcal{C}^{sample}$ and $\mathcal{C}^{basic}$ are in $\mathcal{O}(m \cdot n_S)$ and $\mathcal{O}(\alpha^m)$, respectively. Again, the theoretical cost of $\mathcal{C}^{basic}$ is lower than that of $\mathcal{C}^{sample}$ if $\alpha^m < m \cdot n_S$.

Figure 4.7 illustrates the described approach. There, we derive a bivariate distribution, $x_{1,2}$ (represented through $x_{1,2}^{(10,10)}$) as a joint distribution between x_1 (Gamma distributed with $\alpha = \beta = 2$) and x_2 (normally distributed with $\mu = 0$, $\sigma = 1$). The center of the figure shows the respective histograms $x_1^{(10)}$ and $x_2^{(10)}$, which are accessed to compute inv_{x_1} and inv_{x_2}, respectively. The left-hand side of the figure shows the ACR $\overline{C}^{10}_{T(1),0.8}$. The inversion is illustrated for the single bin $b_{5,8}$ which is marked in the histogram of $\overline{C}^{10}_{T(1),0.8}$. The center

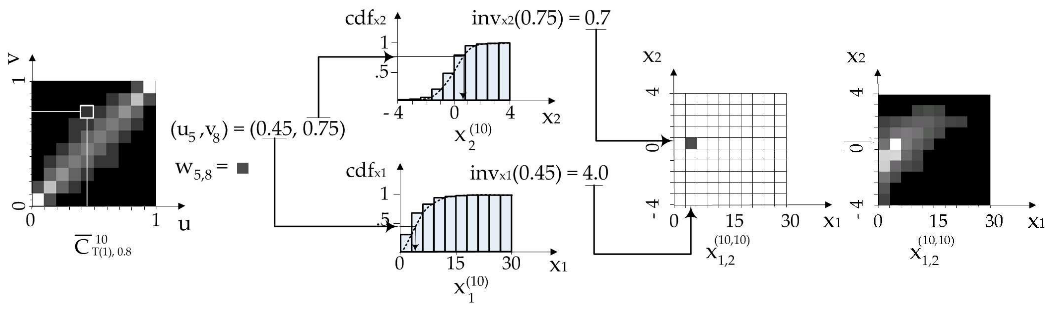


Figure 4.7: Introduction of correlation between univariate distributions

coordinates of $b_{5,8}$ are considered as the coordinates of its representative sample in the unit square, $(u_5, v_8) = (0.45, 0.75)$. Now, we use the density associated with $b_{5,8}$ (represented by a dark shade of gray) as the weight $w_{5,8}$, say, $w_{5,8} = 5$. We now access $x_1^{(10)}$ and $x_2^{(10)}$ to compute the inverse cumulative distribution values of x_1 and x_2 at $u_5 = 0.45$ and $v_8 = 0.75$. This results in the coordinates $x_{1,5} = 4.0, x_{2,8} = 0.7$. Conceptually, those coordinates correspond to a sample in the target joint distribution. The associated weight $w_{5,8}$ is then added to the corresponding bin in the result histogram $x_{1,2}^{(10,10)}$, as represented by the colored bin on the right hand side of the figure. This process is repeated for each of the bins in the ACR, and densities are successively summed up to yield $x_{1,2}^{(10,10)}$, as shown in the far right-hand side of the figure.

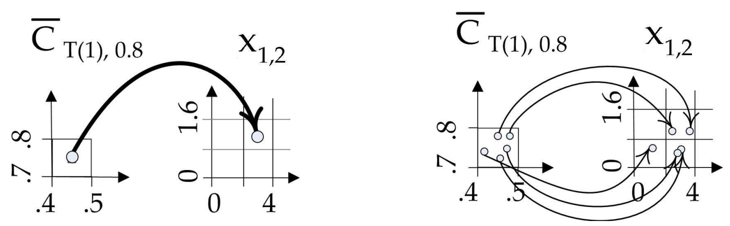
4.4.2 Uniform Spread Approach

ACR Resampling Applying the basic approach implemented by $\mathcal{C}^{basic}$ may result in a large discretization error. This is because for each bin we consider only one sample as its representative, resulting in a total of only α^m weighted samples for the complete ACR. We therefore extend the basic approach by applying a resampling approach to compensate for a part of the discretization introduced in the construction of the ACR. In particular, assuming a uniform spread within each ACR bin $b_{i,j}$, we draw a number of uniformly distributed samples per $b_{i,j}$. This process is implemented by $\mathcal{C}^{resample}$ as shown in Listing 4.2.

Listing 4.2: ACR processing with uniform resampling through $\mathcal{C}^{resample}$

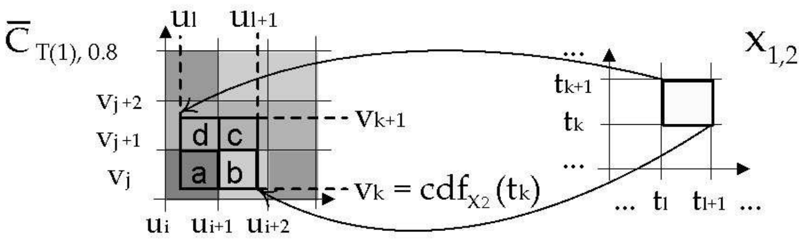
```
$\mathcal{C}^{resample}$($x_1$, $x_2$, $\overline{C}^{\alpha}$):
  for each $b_{i,j}$ in $\overline{C}^{\alpha}$
    for k = 0, ..., $w_{i,j}$
        $u_{i,k}$ = sampleuniform$_u$($b_{i,j}$)
        $v_{j,k}$ = sampleuniform$_v$($b_{i,j}$)
        ($x_{1,i,k}$, $x_{2,j,k}$) = (inv$_1$($u_{i,k}$), inv$_2$($v_{j,k}$))
        add ($x_{1,i,k}$, $x_{2,j,k}$) to $x_{1,2}$
```

The weight $w_{i,j}$ associated with $b_{i,j}$ determines the number of samples to be drawn for $b_{i,j}$. Finally, we invert each of the uniform samples to yield the samples in the result distribution



(a) Basic approach, using $\mathcal{C}^{basic}$

(b) Uniform resampling, using $\mathcal{C}^{resample}$



(c) Reverse processing, using $\mathcal{C}^{reverse}$

Figure 4.8: Alternative inversion methods

$x_{1,2}$. Figure 4.8(b) shows the resulting "dispersion" of the aggregated density value $w_{i,j}$ (e.g., $w_{5,8} = 5$) of an ACR bin $b_{5,8}$ to different locations of the result distribution, while Figure 4.8(a) displays the result of the basic approach, where the complete weight of the ACR bin is assigned to one result histogram bin.

Complexity: For a total of n_{ACR} uniformly drawn samples, the introduction of correlation implies a similar complexity as the "native" sampling-based approach. However, at runtime we save the costs for constructing the copula in the first place. Therefore, assuming we introduce correlation between m distributions $x_j^{(\beta)}$ the induced complexity is determined by n_{ACR} quantile computations in total over all bins for each of the m marginals:

$$\mathcal{C}^{resample}(x_1^{(\beta)}, ..., x_m^{(\beta)}, \overline{C}^{\alpha}, n_{ACR}) \;\in\; \mathcal{O}(m \cdot n_{ACR} \log_2 \beta)$$

The complexity for the histogram construction in the case of $\mathcal{C}^{resample}$ is in $\mathcal{O}(m \cdot n_{ACR})$ and thus, similar to the case of $\mathcal{C}^{sample}$ when $n_{ACR} = n_S$.

Reverse ACR Processing If it is known at the time of correlation introduction (i.e., when $\mathcal{C}$ is executed) that the result distribution will be represented through a histogram with known bin boundaries, we can perform the described approach more efficiently. This assumption generally holds true when we use equi-width histogram representations, which is the case in this thesis. The approach taken to increase the efficiency of $\mathcal{C}^{resample}$ is to *reverse* the mapping between $\overline{C}$ and the target distribution $x_{1,2}$. Consider bins $b_{l,k}, 1 \leq l \leq \beta_1, 1 \leq k \leq \beta_2$ of a target result histogram $x_{1,2}^{(\beta_1,\beta_2)}$, each spanning a rectangle $[t_l, t_{l+1}] \times [t_k, t_{k+1}]$. Now, rather than "dispersing" the weight from each ACR bin $b_{i,j}$ to the result histogram bins $b_{l,k}$, we instead gather fractions of the ACR bin weights for each result bin $b_{l,k}$.

This approach is implemented by the operator $\mathcal{C}^{reverse}$. Figure 4.8(c) illustrates the reverse processing scheme for a single bin $b_{l,k} \in x_{1,2}^{(\beta_1,\beta_2)}$. We compute the density for $b_{l,k}$ as the sum of weights contained in the area of the ACR that maps to the result bin through inversion. To achieve this, we first compute the associated transforms u_l, u_{l+1}, v_k, and v_{k+1}. That is, the boundaries of the result histogram bins are mapped into the unit square covered by the ACR histogram:

$$u_l = cdf_1(t_l), \;\; u_{l+1} = cdf_1(t_{l+1}), \;\; v_k = cdf_2(t_k), \;\; v_{k+1} = cdf_2(t_{k+1})$$

Then, we calculate the fractions of all ACR bins overlapping with $[u_l, u_{l+1}] \times [v_k, v_{k+1}]$. In the figure, we compute f_a, f_b, f_c, f_d as the fractions that the areas marked with a, b, c, and d cover of their associated bins. Finally, we yield the density associated with $b_{l,k}$ as

$$w_{l,k} = f_a \cdot w_{i,j} + f_b \cdot w_{i+1,j} + f_c \cdot w_{i+1,j+1} + f_d \cdot w_{i,j+1} \,.$$

For cases where the number of target bins $\beta_x \cdot \beta_y$ is smaller than the number of n_{ACR}, the reverse scheme is more efficient than the ACR resampling approach due to the decreased number of transformation steps.

Complexity: To build a target histogram with β^m bins using $\mathcal{C}^{reverse}$, we need to compute $m \cdot \beta$ transforms (for each of the β bin boundaries in m dimensions), compute $m \cdot \alpha\beta$ overlap fractions (of which most can be omitted since they evaluate to zero trivially), and finally compute the sum of fractions for each of the β^m result bins. The latter step dominates the overall complexity such that

$$\mathcal{C}^{reverse}(x_1^{(\beta)}, ..., x_m^{(\beta)}, \overline{C}^{\alpha}) \;\; \in \;\; \mathcal{O}(\beta^m)$$

Again, under the assumption that $\beta^m m \cdot n_S$, the reverse approach implies lower costs than the application of $\mathcal{C}^{sample}$.

4.5 Query Processing

In Figure 4.9 we can see how the overall process for introducing a correlation structure proceeds. Generally, one can either use precomputed ACRs that are accessed from the ACR store or ACRs extracted from historical data at query time.

From a system point of view, the ACR store containing all available ACRs in a dedicated histogram table as tuples (id, b, w) associating the ACR id with its bins b and the density w per bin. Those can be either precomputed ones as well as ACRs previously empirically derived and persisted. An additional index associates, for each precomputed ACR $\overline{C}_{H,d}$, the correlation parameters H and d with the id of the respective ACR. For selecting an ACR, a user can either provide the assumed correlation parameters based on expert knowledge, or they can be derived through rules at the application-level. For example, for modeling high joint tail dependencies, a rule could imply that $H = T(1)$, while the requirement for an asymmetric correlation structure could yield to the selection of an ACR computed from a Clayton copula.

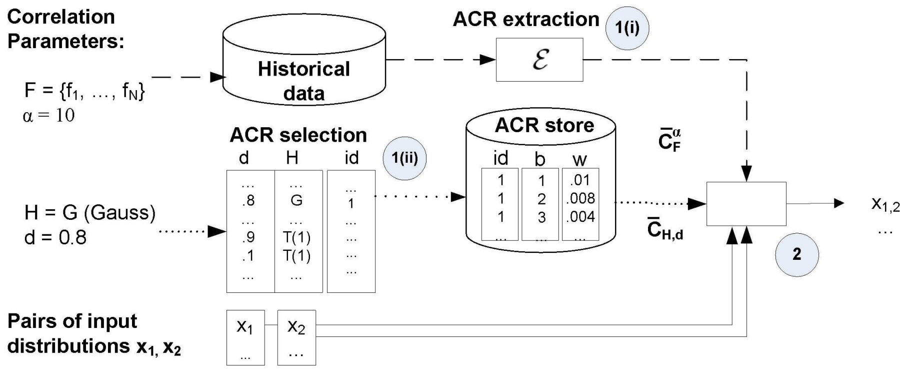


Figure 4.9: Overview of process for correlation introduction

Listing 4.3 illustrates queries for introducing correlation between two distributions x_1, x_2 (x1,x2).

Using Query (i), a user can first derive an ACR (acr) empirically from selected facts (F). Then, the derived copula is applied for correlation introduction, using one of the variants of $\mathcal{C}$. Alternatively, the user can provide parameters H and d for selecting a pre-computed ACR from the store, as shown in Query (ii). In Figure 4.9, the two alternatives are indicated by dashed arrows for (i) and by dotted arrows for (ii).

Listing 4.3: Correlation queries

```
(i)
WITH acr AS (SELECT Empirical(F) FROM F)
  SELECT Correlate(acr,d1,d2)
    FROM dists d1, dists d2
   WHERE d1.id = dx AND d2.id = dy

(ii)
SELECT Correlate(acr,d1,d2)
  FROM dists d1, dists d2, acr
 WHERE d1.id = dx AND d2.id = dy
   AND acr.H = H AND acr.d = d
```

Having received either of Query (i) or Query (ii), the method proceeds through two steps:

1. Retrieve the ACR by (i) extracting $\overline{C}_F^\alpha = \mathcal{E}(F, \alpha)$ from facts F or (ii) selecting the ACR $\overline{C}_{H,d}$ from the ACR store.

2. Introduce the correlation structure using one of the available procedures for $\mathcal{C}$, namely
    a) if $x_{1,2}$ shall be represented by a histogram $x_{1,2}^{(\beta_1,\beta_2)}$ with precomputed bin boundaries, apply $\mathcal{C}^{reverse}$
    b) else, apply $\mathcal{C}^{basic}$ or $\mathcal{C}^{resample}$

Using a selected or extracted ACR, the $\mathcal{C}$ operator can correlate one or more distribution pairs ($x_1 \in X_1$, $x_2 \in X_2$), accessing the respective ACR only once and processing it for each pair of marginal distributions. This is, for example, relevant when we want to apply the same correlation structure to many assets in a portfolio. If an empirical copula $\overline{C}_F^{\alpha}$ shall be used, it is first computed over the selected fact data F using $\mathcal{E}$, and then used in the correlation process, applying $\mathcal{C}(\overline{C}_F^{\alpha}, x_1, x_2)$. Empirical copulas can also be persisted in the ACR repository for later reference through an associated id, similar to the precomputed ACRs.

Generally, depending on the overall analysis process, a produced joint distribution $x_{1,2}$ can then be used either directly as a result, stored for further access (as a histogram $x_{1,2}^{(\beta_1,\beta_2)}$), or processed by subsequent operators.

The presented functionality shall enable users to carry out fast interactive analyses in decision making processes. Fortunately, ACRs provide some beneficial characteristics that open up opportunities for optimizing the required computations, as described subsequently.

4.5.1 Optimizing Query Plans

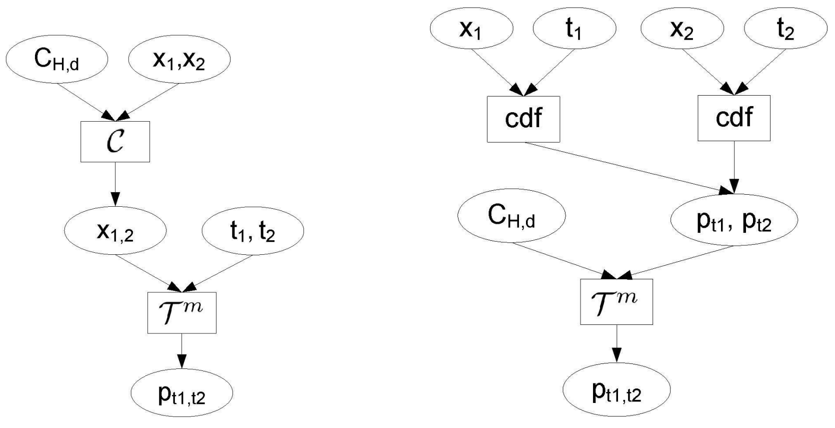
A first optimization consists in the rewriting of queries that involve the computation of $\mathcal{T}^m$ or $\mathcal{M}$ preceded by $\mathcal{C}$.

As an example, consider the application of $\mathcal{M}$ in Task B.1, where we want to analyze $x_{f,h}$ by computing a marginal $x_{f|h>\$7\,bn}$, and the risk that both f and h will exceed specified thresholds $t_f = t_h = \$15\,bn$ under the assumed correlation.

Marginal Distributions When computing the marginal distribution using the processing procedure outlined above, we would first apply $\mathcal{C}$ to calculate $x_{f,h}$ and then apply $\mathcal{M}$. At this point, assuming we do not need the complete distribution $x_{f,h}$ for further steps, this execution strategy induces a large number of unnecessary calculations—only a part of $x_{f,h}$ (i.e., only the region where $h > \$7\,bn$) will contribute to $x_{f|h>\$7\,bn}$. To prevent overhead and only compute the information necessary for further steps, we can exploit the feature that the ACR $\overline{C}_{H,d}$ reflects a *mapping* from the unit interval to the concrete marginals x_f, x_h; hence, we can in turn map the relevant part of $x_{f,h}$ to a subregion of the applied ACR to access and process this ACR subregion only. We can easily determine this region based on the condition predicates ω_j of $\mathcal{M}$. Specifically, for a condition $\omega_{op}(t_x), op \in \{<,>\}$ we only need to consider the ACR region that satisfies $v \; op \; p_{t_x}$, with $p_{t_x} = cdf_x(t_x)$, when applying $\mathcal{C}$. As a result of the decreased number of required inversions, the runtime costs for the correlation step decrease almost linearly by a factor of $1/(p_{t_x})$ or $1/(1-p_{t_x})$ for *op* being $<$ or $>$, respectively.

EXAMPLE: In Task B.1, to compute the marginal $x_{f|v_h}$ with $\omega_h(7\,bn, \infty)$ directly over the result of $\mathcal{C}$, we push down the selection of the relevant region of $x_{f,h}$ (i.e., $h > \$7\,bn$) to select the region of the applied ACR where $v > cdf_h(\$7\,bn)$.

Joint Tail Probability Moreover, when applying $\mathcal{T}^m$ to compute a joint tail probability $p^u_{t_f,t_h}$, we can achieve an even greater optimization effect. This optimization approach is depicted in the two alternative query graphs in Figure 4.10. The graphs illustrate how the result of $\mathcal{T}^m$ can be computed *directly* on $\overline{C}_{H,d}$ rather than using $\mathcal{C}$ to process a region of the ACR and introducing a correlation in a first step. As a consequence, we do not need to apply $\mathcal{C}$ at all. Instead, we compute the uniform transforms of t_f and t_h, $p_{t_f} = cdf_{x_f}(t_f)$ and $p_{t_h} = cdf_{x_h}(t_h)$ and use them as parameters to the $\mathcal{T}^m$ operator, which we now apply directly to $\overline{C}_{H,d}$. That is, we compute $\mathcal{T}^m(\overline{C}_{H,d}, p_{t_f}, p_{t_h})$.



(a) Initial query graph (b) Optimized processing

Figure 4.10: Alternative query graphs for calculating p_{t_1,t_2}

Note that the described approach relates to the reverse ACR processing scheme described in Section 4.4.2; conceptually, we now determine the region of $\overline{C}_{H,d}$ associated with one "tail bin" and aggregate the density contained therein. Since this method implies only two computations of *cdf* followed by an aggregation of the relevant ACR bin weights, we achieve a large efficiency gain, as shown in Section 7.5.3.

It is noteworthy that the optimizations described above rely on the fact that ACRs can be accessed as precomputed structures. Thus, we can restrict the computations that are executed when applying either $\mathcal{M}$ or $\mathcal{T}$ to parts of the precomputed correlation information. Using the sample-based implementation of $\mathcal{C}$, does not allow for using those optimization approaches directly. In MC-based solutions, one could support the efficient use of copulas by means of techniques for sampling from the tails of joint distributions, such as discussed in [JXW$^+$10].

4.5.2 Parallelization

An additional, orthogonal approach to optimize the correlation processing stems from the fact that many calculations can be executed independently on the individual bins of an ACR, and therefore are amenable to parallelization.

Correlation Introduction Since $\mathcal{C}$ can process a subset of the α^2 ACR bins independently from any other subset, we can parallelize its execution. This applies both to the basic approach implemented by $\mathcal{C}^{basic}$ and the resampling approach implemented by $\mathcal{C}^{resample}$. Using t threads, each can be assigned a t^{th} subregion of the complete ACR to produce a partial result distribution. Those t partial results are then merged into the final distribution $x_{1,2}$. The merging step consists in a bin-wise addition of bin frequencies, which is a simple operation and therefore implies only slight computational overhead. Similarly, we can use t threads to compute the reverse ACR processing scheme as implemented by $\mathcal{C}^{reverse}$, where each of the t threads can be assigned with a t^{th} part of the bins of the target result histogram.

In this case, the use of parallelization pays off foremost when the target histogram has a high granularity, i.e., is represented by a relatively large number of bins β_1 and β_2 (e.g., $\beta \geq 100$). A similar benefit in performance can be gained when using fine-granular or higher-dimensional ACRs (e.g., $\alpha \geq 100$, or $m \geq 3$). This is because in those cases we have a relatively high computational effort due to the large number of overlap fractions that must be computed, as opposed to the relatively low overhead induced by the synchronization and merging steps of the parallelized version of $\mathcal{C}$.

Empirical Copula Derivation We can also parallelize most steps of the empirical copula derivation operator, $\mathcal{E}$, i.e., computing the marginal distributions over the $|F|$ samples and computing the uniform transforms for each sample value $(f_{1,i}, f_{2,i})$. In the parallelized implementation of the operator $\mathcal{E}$, each of the t threads first computes marginal distributions over a t^{th} subset of the samples which are then merged into the complete marginal distribution histograms. Then, each thread transforms a t^{th} part of the samples based on the computed marginal distributions. Then, t partial ACR histograms can be constructed over the samples, which are finally merged into the result ACR.

4.6 Summary

This chapter discussed means for the flexible and efficient representation and processing of arbitrary correlation structures at the database. The primary goal of the proposed techniques was to enable users to perform flexible analyses of joint distributions under potentially different correlations. To this end, it is necessary to allow them to introduce arbitrary correlation structures between arbitrary distributions. This functionality was addressed using the concept of copulas. First, the copula approach for correlation representation was discussed. Then, we considered an adaption of this approach to the database by means of the ACR concept. Further, it was pointed out how a user can either use precomputed ACRs in cases where only a purely *assumed* correlation shall be evaluated, or first extract ACR structures from existing data. Then, the functionality for introducing correlation to data in the analysis process was presented and different alternative versions of the correlation operator $\mathcal{C}$ were proposed. Finally, opportunities for optimization through query rewriting and parallelization were pointed out.

In Chapter 7 the application of the proposed operators in an implementation of the discussed use case are demonstrated. Also, the efficiency and accuracy of the proposed approaches are evaluated. Further, we consider an exemplary comparison with a sample-first approach in order to point out the applicability of the generic copula technique, as well as investigating the benefits of the ACR-based approach.

After having presented the provided set of operators for scenario analysis in the last two chapters, the next Section addresses a crosscutting aspect of the scenario analysis process. We consider the question how the iterative nature of the analysis process can be supported. The focus there is to provide a methodology for an efficient recomputation of scenarios based on changes in input data or assumptions. This methodology particularly exploits *provenance* information captured during an analysis process. Among other optimizations, the captured information enables the rewriting of query plans that involve subsequent applications of multivariate analysis operators, as has been discussed above.

5 Scenario Recomputation

The computation of scenarios based on various assumptions can initially offer valuable insights to the business. However, in order to allow for an interpretation of the derived scenarios, it is equally important to enable the user to track the applied analysis steps. For example, when analyzing sales projections, such as in the scenarios derived in Use Case 1, first of all the user needs to know about the assumptions that were made with respect to the factors that influence the future sales. Only based on this knowledge the user will be able to reason about (and compare) different scenarios. Second, when actual data contradicts an applied assumption (e.g., the actual economy turns out worse than expected) the user wants to reflect the plan-actual deviations in any results derived from the assumption. That is, he needs to be able to recompute the analysis results based on available information about changes in input data. Similarly, the user requires means to reflect *assumed* deviations in the underlying inputs.

Those two aspects relate to the general tasks of (i) querying data provenance backwards (to investigate the data derivation) as well as forwards (to find dependent data) and (ii) investigate the impact of changes in base data on dependent data by re-evaluating underlying queries efficiently. The following chapter will focus on the latter aspect; the former serves as a prerequisite to investigate relevant input data and operators that must be recomputed in a re-evaluation of the query.

5.1 Provenance and Recomputations in the Analysis Process

First, we briefly consider the requirements for provenance handling and efficient recomputation that are addressed in the scope of this chapter. To this end, consider an example analysis process for one of the use cases discussed throughout the thesis, which can now be implemented using the analysis functionality presented in the previous two chapters.

5.1.1 Example Analysis Process

Recall Use Case 1, Task A, where a company plans a marketing campaign to optimize the outcome on the generated sales revenues of their to-be-launched product $P2$. Within the campaign, various marketing measures shall be applied to address the customers, such as mailings, vouchers, or TV ads, each causing different expenditures per customer . The marketing manager wants to forecast and analyze the effectiveness of different marketing expenditures (more precisely, different distributions of per-customer expenditures) with regard to the monthly sales generated per customer. Moreover, the manager may, for example, aggregate the projected sales values to yield the total expected sales values per product group, location, etc.

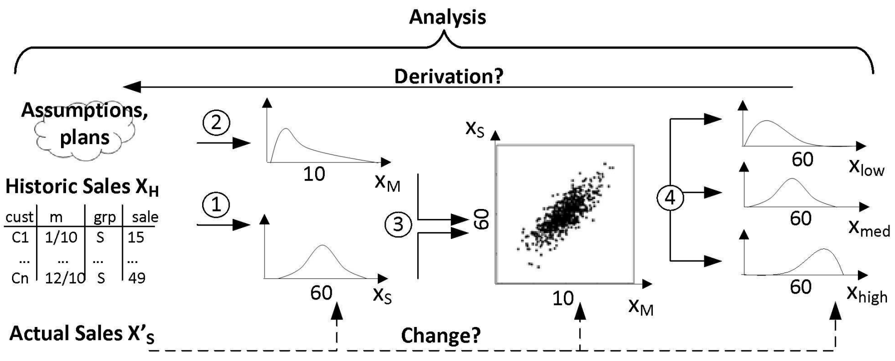


Figure 5.1: Analysis process of Use Case 1: Sales planning conditioned on marketing expenditures

Figure 5.1 shows the analysis process for the described task: ① As a starting point, the user derives a series of forecast values for the monthly per-customer sales amount (x_S) of $P2$. For simplicity, he computes those forecasts as distributions from the historic monthly sales data of the reference product $P1$ in the previous year (the set of monthly sales amounts, $\dot{X}_H$) by applying $\mathcal{D}$[1]. ② He also provides information about the (planned) distribution of monthly per-customer marketing expenditures (x_M). For example, he may plan a high number of low-cost activities (say, for newsletters, and direct mailings), as well as higher costs incurred by vouchers and promotions (indicated by a Gamma distribution in the Figure). ③ To introduce the assumption that sales will exhibit a high (positive) dependence on the marketing expenditures, the $\mathcal{C}$ operator is applied, which computes the joint distribution between x_S and marketing costs, x_M. ④ Finally, the user wants to investigate the distribution of monthly sales in the cases of low-, medium-, and high marketing expenditures (e.g., for per-customer expenditures within $low = [\$0, \$5]$, $med = [\$5, \$10]$, and $high = [\$10, \infty]$, respectively), which is derived using $\mathcal{M}$ and results in three alternative scenarios denoted as x_{low}, x_{med}, and x_{high}. Note that in step①, the user may naturally also derive multiple monthly forecast values, e.g., for various products, regions, etc; this aspect is omitted in the illustration for reasons of simplicity. Still, it is addressed through the capture and processing of provenance information for datasets within this chapter.

Recomputation A user should be able to recompute the final results (in this case, x_{low}, x_{med}, and x_{high}) after changes in (fractions of) the input data have occurred. Such recomputations should naturally be performed *efficiently*, i.e., with minimal overhead with respect to the proportion of the changes in data.

[1]For simplicity, we omit the step of selecting and aggregating historic data assuming they are readily available.

In the exemplary analysis process for Use Case 1, assumptions about per-store sales (x_S) may become invalid due to deviating actual sales data ($x_{S'}$). For example, a shift in per-customer expenditures may lead to a situation where the recorded sales x_M exhibit a distribution $x_{S'}$ with mean $\$50$ rather than the initially assumed mean of $\$60$. This would render the assumed distribution for x_S (see step ① in Figure 5.1) increasingly inconsistent. Alternatively, the user may want to modify the sales forecast values to reflect *assumed* deviations.

Similar to the described manual approach to recomputation (i.e., deviations in inputs are specified by a user or derived over new evidence data), an automatic procedure could recompute results over *several variations* of input data. From those scenarios, a user can then for example identify relevant (critical or promising) "neighboring" scenarios of an initial scenario. In this case, the need for efficient incorporation of changes becomes even more apparent since many new computations must be performed. In the remainder of this chapter, we investigate how we can exploit specifics of the underlying data and the class of the analysis functionality discussed in Chapters 3 and 4.

5.1.2 Data and Operator Characteristics

In the scope of this work, the central data and operator characteristics with regard to the recomputation of analyses are as follows:

Deviations in Continuous Input Distributions When recomputing analysis results based on deviations in continuously distributed input data, we need to consider the deviations of an old and new input value over the whole support interval of its distribution. This is in contrast to approaches for recomputation and sensitivity analysis in the context of discrete probabilistic input data, where the deletion or updating of a discrete value or tuple probability is considered. In Section 3.2.1 the representation of delta information Δ_x, represented by a delta histogram Δ_x^H between two distribution representations x and x', has been described. This representation will be used to reflect deviations in the course of recomputation.

Focused Class of Operators Both the application of relational operators such as selection and grouping as well as the previously discussed statistical operators are relevant within the analysis process. In this chapter, similar to the previous chapters, we explicitly focus on the application of analytic operators, and specifically on the capture and exploitation of specific provenance information that can be exploited in the recomputation process. In particular, we do *not* address the introduction and evaluation of dependencies that are introduced through relational queries including joins over probabilistic attributes.

Clearly, relational operators are applied in all but a few cases before any statistical analysis operators are used. For example, before deriving distributions over fact data, the relevant data is first selected based on a selection predicate, and possibly grouped, e.g., per product. Capturing and querying the provenance of or dependencies in data derived through relational queries, and their (selective) recomputation based on changes in input data has been the topic of previous research such as [ABS$^+$06, SMS$^+$08, ISW10]. The focus of the subsequently presented approaches is explicitly on a complementary methodology that specifically targets the efficient recomputation of analytic operators. The central approach is to exploit specific provenance information captured during the initial evaluation phase to perform recomputation steps only over those deviations in input data that can indeed *impact* the (intermediate) computation results.

5.1.3 Provenance for Scenario Data

Scenario data reflect potential realizations of the future, and thus involve assumptions about influences (drivers) of future developments. Those may change or need to be adapted throughout time. Our focus is to support a continuous, iterative scenario derivation process. An import aspect therein is to allow users to consider the evolving state of underlying data, i.e., to incorporate changes in input data on derived analysis results.

Tracing and Querying the Scenario Derivation As a first step to enable the analysis and incorporation of changes, a user must first know about the derivation of scenario data. For example, a user reviewing potential sales needs to know about underlying assumptions about the marketing expenditures and correlation with sales; in case the planned sales numbers seem exceedingly low, or deviate strongly from actual recorded sales, he may investigate whether this stems from inappropriate assumptions, or rather is an indication for marketing optimization potential. Once the user knows about relevant base data, he can analyze the impact of changes in the base data on the overall result of the analysis.

Change Impact Evaluation To incorporate actual or assumed deviations from applied hypotheses in their analyses, users need to evaluate how they impact previously derived data. For example, deviations from prospected customers sales patterns may influence computed revenue scenarios. A shift in per-customer expenditures may lead to a situation where sales S are distributed with mean $\$50$ rather than a forecast mean of $\$60$, possibly rendering an assumed distribution for S (see step ① in Figure 5.1) increasingly inconsistent. In general, uncertainty can change over time and assumptions can get inconsistent as actual data is obtained. A user must be able to query how derived scenarios may be affected, and possibly recompute them based on the identified deviations.

We enable users to incorporate deviations in two ways:

Exact Recomputation To incorporate *exact* information about deviations in base data, the user can recompute all affected intermediate data of an analysis process based on the associated delta information, represented by Δ^H as described in 3.2.1.

Approximate Recomputation and Deviation Analysis In other cases users may not know about exact changes, but rather need to investigate the effect of *assumed* deviations. Then, an alternative approach is to evaluate change impacts in an approximate fashion based on a functional deviation approximation, denoted by Δ^T.

To support those tasks in an efficient manner, we need to consider both data-related and process-related aspects of the analysis. In Section 5.2 the capture of provenance information regarding both aspects is described. Specific knowledge about characteristics of data dependencies is incorporated with information about operator applications and their data in- and output, which is captured through a graph representation. In Section 5.3 we discuss the recomputation of analysis results using exact and approximate deviation representations, exploiting the recorded provenance information.

5.2 Scenario Provenance Capture

When computing scenarios, we need to keep provenance information about both the transformations and data involved in the process to enable users to track how analysis results were derived and to recompute them. Moreover, for an *efficient* evaluation of change impacts *specific* information about applied operations is required.

As stated above, rather than considering changes in probabilities of discrete tuple alternatives (as for deletion propagation [GKT07] or sensitivity/explanation analysis [KLD11]), the continuous case necessitates that we address the impact of changes in (*parts* of) the distributions represented by data items. This impact clearly depends on the applied operator o. After describing a basic provenance graph that captures data derivation on the high level of operator applications and datasets, we address the capture of operator-specific provenance information both on the level of data sets and the detailed level of data items.

5.2.1 Basic Provenance Graph Structure

Formally, we capture the *transformation provenance* of a scenario computation through a graph $\mathcal{G} = \{N, E, ann\}$, where

- N is the set of nodes, $N = N_O \cup N_D$. Operator nodes $o \in N_O$ represent processing steps, and data nodes $d \in N_D$ represent data artifacts, i.e., data items or sets.
- $E \subseteq N \times N \times T$ is the set of typed edges representing provenance relations, where T is the set of relation types.
- $ann : N \cup E \rightarrow A(K \times V)$ associates a node $n \in N$ or edge $e \in E$ with additional attribute information stored by key-value pairs (k, v), such as the name of the roles associated with an edge.

In the following, we use the notation $n_1 \,{}_{(r_1)}\!\xrightarrow{t}_{(r_2)}\, n_2$ for an edge of type t that connects node n_1 to n_2, where labels r_1 and r_2 *may* be used to specify the role of n_1 and n_2. We will exemplify the use of roles below; for better readability we omit them whenever they are not required. When describing provenance, we refer to the underlying data x or X through its associated data node d. We use edge types $T = \{used, wgb, wdf\}$, similar to the provenance relations from the OPM [MFF$^+$07]. An edge $d \xrightarrow{wgb} o$ reflects that (the item x associated with) d *was generated by* the operator o while $o \xrightarrow{used} d$ means that o used (the item x associated with) d as input. Further, $d_1 \xrightarrow{wdf} d_2$ states that d_1 *was derived from* d_2; it can be inferred from edges $d_1 \xrightarrow{wgb} o$ and $o \xrightarrow{used} d_2$[2]. We store instances of each node and edge type, including associated annotations, in dedicated provenance relations, as exemplified below.

EXAMPLE: Figure 5.2 shows a basic provenance graph for the derivation of x_{low} in the analysis process above. The graph captures the application of operators $\mathcal{D}$, $\mathcal{C}$, and $\mathcal{M}$. For example, nodes d_M and d_S represent the planned marketing expenditures (x_M) and sales forecasts (X_S) while the edges $d_S \xrightarrow{wgb} o_{\mathcal{D}}$ and $o_{\mathcal{D}} \xrightarrow{used} d_H$ represent the derivation of sales forecasts $x_{S,i} \in X_S$ from $\check{X}_H$ through the operator $\mathcal{D}$. The application of the $\mathcal{C}$ and $\mathcal{M}$ operators are similarly reflected with their associated input and output data nodes. The derivation of x_{med} and x_{high} are similar, differing only in the *used* input nodes of $\mathcal{M}$. Table 5.3 shows the relations that storing the graph structure containing data nodes (5.3(a)) and edges (5.3(b)).

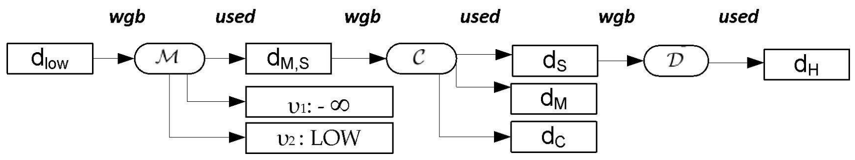


Figure 5.2: Basic use case provenance graph

data nodes		
d	x	ver
d_H	$\check{X}_H$	v_0
d_S	X_S	v_0
d_M	x_M	v_0
$d_{M,S}$	$X_{M,S}$	v_0
d_C	x_C	v_0
d_{low}	X_{low}	v_0
d_υ	LOW	v_0
d_{op}	$>$	v_0
d_H	$\check{X}_A$	v_1
d_S	X'_S	v_1
d_{low}	X'_{low}	v_1

(a) Data nodes d

wgb-edges		
d	o	type
d_S	$\mathcal{D}$	wgb
$d_{M,S}$	$\mathcal{C}$	wgb
d_{low}	$\mathcal{M}$	wgb

used-edges		
o	d	type
$\mathcal{D}$	d_H	$used$
$\mathcal{C}$	d_M	$used$
$\mathcal{C}$	d_S	$used$
$\mathcal{C}$	d_C	$used$
$\mathcal{M}$	$d_{M,S}$	$used$
$\mathcal{M}$	d_{op}	$used$
$\mathcal{M}$	d_υ	$used$

(b) Edges $e \in E$

Figure 5.3: Provenance relations

[2]Note that we only adopt relation types from OPM and do not account for subtle semantic aspects of the OPM relations.

5.2.2 Scenario Versions and Modification

Given changes in input data, a user could in practice directly replace the old input values and evaluate the effect of those updates on previously computed analysis results. However, in many cases, the existing (derived) scenario data must not be changed directly, even if new evidence renders them invalid. The data may be integrated in business reports, or otherwise must be kept available for reference. Therefore, we handle information about changes as delta information, storing different versions of both data items and provenance information.

Scenario Version Provenance

Alternative scenario versions are created when results are recomputed based on changed input data. A version identifier *v* distinguishes provenance elements for different scenario versions, yet retaining the relation between alternative computations. Since alternative versions of a scenario computation may differ only in a small number of inputs, we store the provenance graph for a version v_{i+1} as *delta* to the graph $\mathcal{G}$ of the previous version, v_i instead of storing a complete new provenance graph. This means that we add only those provenance elements which distinguish v_{i+1} from v_i. In case of small differences, this approach is more space-efficient than storing the complete new provenance $\mathcal{G}'$, at the cost of re-assembling $\mathcal{G}'$ when it is queried. See [ZDH^{+}11] for a discussion of the trade-off between runtime and storage efficiency resulting from this approach.

As an example, Table 5.3(a) shows the data nodes d_H, d_S, and d_{low}, distinguished through version $ver = v_0$ and v_1. The v_1-nodes are added when the scenario is re-computed based on *actual* sales data $\dot{X}_A$, with (d_S, v_1) and (d_{low}, v_1) representing the derived data X'_S and X'_{low}, respectively. In the remainder of the chapter, we will implicitly refer to two successive versions of data associated with a node d as items x and x'.

Deviation Information Δ_x

Recall that modifications to data are stored as *delta* information $\Delta_{x,x'}$ between two items x and x', denoted as Δ_x if the context (i.e., the relation between two successive versions of a data item, x, x') is clear. In particular, *delta histograms* Δ_x^H, which are computed from the differences of cdf_x and cdf'_x, are stored to capture deviations between two items, from which the deviation at any position v can be computed as $\Delta_x^H(v)$ in constant time, similar to the computation of cdf_x on a regular distribution.

Deriving Δ_x^H If a distribution y was initially created through $\mathcal{D}$ from fact data $\dot{X}$ and new facts have been recorded (or old ones updated), it may become necessary to recompute y' based on the new (partially deviating) fact data, $\dot{X}'$. Under the working assumption that each $\dot{x}_i \in \dot{X}$ can be associated with its corresponding $\dot{x}_i{}' \in \dot{X}'$ (e.g., through a unique sales id), the deviation information Δ_y^H can be computed based on the differences of the two sets. That is, we define sets $\dot{X}^{\Delta} = \{\dot{x}_i | \dot{x}_i \notin \dot{X}'\}$ and $\dot{X}'^{\Delta} = \{\dot{x}'_i | \dot{x}'_i \notin \dot{X}\}$ and compute two delta histograms, $\Delta^{H-} = \mathcal{D}(\dot{X}^{\Delta}, \beta)$ and $\Delta^{H+} = \mathcal{D}(\dot{X}'^{\Delta}, \beta)$. Then, Δ_y^H is built by bin-wise subtraction and addition of those deltas, to yield $\Delta_y^H = \Delta^{H+} - \Delta^{H-}$, which can then be accessed when recomputing scenarios that use y as input data.

Δ_x-based Selective Recomputation The information encoded in Δ_x^H can now be used directly to re-compute the value $\dot{p}' = cdf_{x'}$ based on a known (initial) result $\dot{p} = cdf_x$. More specifically,

$$\dot{p}' = \dot{p} + \dot{p}^{\Delta} \quad \text{with} \quad \dot{p} = cdf_x(\upsilon) \quad \text{and} \quad \dot{p}^{\Delta} = \Delta_x^H(\upsilon)\,.$$

Opposed to that, the computation of $inv_{x'}$ requires us to always access the complete information of x', i.e., both the base and the delta information associated with x'. Thus, the general aim is to only selectively recompute operators that rely on inv based on a *previous computation* of Δ_x^H.

The use of deltas for deviation capture and the result recomputation based on such delta information are discussed in Section 5.3.2, which also addresses the case of using *approximate delta representations* Δ^T. In order to exploit the deviation information to evaluate change impacts, it is however first necessary to extend $\mathcal{G}$ by capturing operator-specific provenance information. This includes the capture of specific edges at the operator level, as well as the construction of predicates over the delta information Δ_x on the level of individual data items.

5.2.3 Representing Modeling Assumptions

First, we capture information about related *components* of data items in the analysis process. The intuition is that, by applying selected operators, we introduce assumptions about correspondences between components of the input and the output data items. For example, when correlating two distributions x_1, x_2 to yield a joint distribution y, this means that x_1 and x_2 model the first and second dimension component of y. This information goes beyond the semantics of a mere input-output dependency, essentially reflecting the composition of multidimensional distributions, similar to probabilistic graphical models. Thus, it is captured through a dedicated edge type denoted by $\approx$, i.e., we extend $\mathcal{G}$ such that $\mathcal{G}.T = \mathcal{G}.T \cup \{\approx\}$.

$$d_{in}\ {}_{r_{in}}\!\xrightarrow{\approx}_{r_{out}}\ d_{out}$$

An $\approx$-edge is added to $\mathcal{G}$ and stored in a dedicated provenance relation depending on the applied operator. Attached roles r_{in} and r_{out} refer to the affected dimension components of the (multivariate) distributions associated with the source and target node, respectively. The $\approx$- edge is transitive, such that two edges $d_1\ {}_{(r1)}\!\xrightarrow{\approx}_{(r2)}\ d_2 \wedge d_2\ {}_{(r2)}\!\xrightarrow{\approx}_{(r3)}\ d_3$ imply $d_1\ {}_{(r1)}\!\xrightarrow{\approx}_{(r3)}\ d_3$.

Table 5.2 formally lists the edges recorded for the multivariate operators.

EXAMPLE: Table 5.1(a) shows the $\approx$-edges introduced for our use case. For a bivariate distribution $x_{M,S}$ resulting from $\mathcal{C}(x_M, x_S, x_C)$, we know that x_M and x_S model the *dim*-components of $x_{M,S}$, and x_C models the cor-component, respectively. Further, the dimension component dim_2 of $d_{M,S}$ corresponds with the resulting (conditional) marginal distributions associated with d_{low}, d_{med}, and d_{high}. Figure 5.4 illustrates part of the resulting graph structure (omitting the derivation of nodes d_{med} and d_{high} for simplicity).

(a) $\approx$ edges

ver	d_{in}	d_{out}	r_{in}	r_{out}
v_0	d_M	$d_{M,S}$	—	dim_1
v_0	d_S	$d_{M,S}$	—	dim_2
v_0	d_C	$d_{M,S}$	—	cor
v_0	$d_{M,S}$	d_{low}	dim_2	—
v_0	$d_{M,S}$	d_{med}	dim_2	—
v_0	$d_{M,S}$	d_{high}	dim_2	—

(b) $\hookrightarrow$ edges

ver	d_{in}	d_{out}	r_{in}	Impact predicatesϕ
v_0	$d_{M,S}$	d_{low}	dim_1	$\phi_l : (\Delta_{x_{M,S}.dim_1}(v) \neq 0)$, $v <$ LOW
v_0	$d_{M,S}$	d_{med}	dim_1	$\phi_m : (\Delta_{x_{M,S}.dim_1}(v) \neq 0)$, HIGH $> v >$ LOW
v_0	$d_{M,S}$	d_{high}	dim_1	$\phi_h : (\Delta_{x_{M,S}.dim_1}(v) \neq 0)$, $v >>$ HIGH

Table 5.1: Operator-specific edges

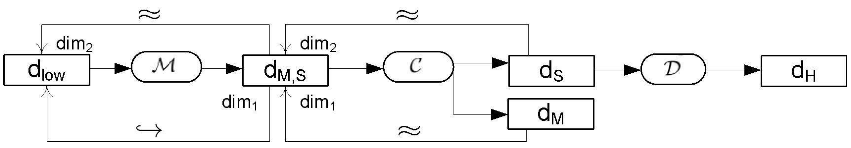


Figure 5.4: Operator-specific provenance edges for the example analysis

5.2.4 Representing Operator Impact

The results of some operators depend on a selected part of x only. This part is determined either by applied predicates ω or processing parameters such as the quantile value used in the computation procedure of $\mathcal{A}_{MAX/MIN}$. In such cases, only deviations in the relevant part of x impact dependent data y, or, are *impact-relevant* for y. To incorporate information about the impact of a deviation Δ_x on y, $\mathcal{G}$ is extended by an additional edge type $\hookrightarrow$, i.e., $\mathcal{G}.T = \mathcal{G}.T \cup \{\hookrightarrow\}$:

$$d_{in}\ {}_{r_{in}}\!\overset{\hookrightarrow}{\longrightarrow}_{r_{out}}\ d_{out}$$

The $\hookrightarrow$-edges are annotated with either a single *impact predicate* ϕ or a collection of impact predicates, Φ, where each $\phi_i \in \Phi$ is associated with an item $x_i \in X$ at the source node d_{in} of the $\hookrightarrow$-edge. That is, while $\approx$- and $\hookrightarrow$-edges are captured on the level of operator nodes, predicates ϕ are captured also on the level of data items, thus representing a more fine-grained information about the computation of data items at the target data node d_{out}. The evaluation of ϕ determines whether a deviation Δ_x at node d_{in} potentially can impact the data associated with d_{out}. That is, only if ϕ evaluates to *true*, the corresponding data item must be considered in the recomputation. For an operator o we construct ϕ from its specific parameters, such as applied predicates ω. In some cases, predicates ϕ_i additionally reference (intermediate) processing results for each x_i, such as the selection probabilities p_i in the case of σ^{τ}. Table 5.2 lists the operator-specific $\hookrightarrow$-edges and impact predicates which are briefly explained in the following.

For $\mathcal{T}(x, \omega)$, we record $d_{in} \overset{\hookrightarrow}{\longrightarrow} d_{out}$, annotated with predicate $\phi : (\Delta_x(\upsilon_1) \neq 0 \vee \Delta_x(\upsilon_2) \neq 0)$ since for an applied predicate $\omega(\upsilon_1, \upsilon_2)$ any deviation of data associated with d_{in} at positions υ_1 or υ_2 will most likely impact the result at d_{out} (i.e., potentially yield a different tail probability). In the case of $\mathcal{Q}$, a deviation Δ_x impacts the result at d_{out} when $\Delta_x(\dot{y}_q) \neq 0$, i.e., when there is a deviation at the position of the previously computed quantile value $\dot{y}_q$. For $\mathcal{D}$, $\mathcal{A}_{SUM}$, and $\mathcal{C}$, *any* deviation in input data leads to a change in d_{out}; therefore, we keep no explicit $\hookrightarrow$-information for those operators. Similarly to the one-dimensional case, for $\mathcal{T}^m$ we record predicates ϕ_{dim_j} for deviations in each dim_j–component of the input distribution. For the operator $\mathcal{M}$, the predicates ϕ_{dim_j} are associated with the dimensions that are marginalized out; each ϕ_{dim_j} evaluates to *true* if there is a deviation $\Delta_{x.dim_j} \neq 0$ anywhere in the interval $[\upsilon_1, \upsilon_2]$.

EXAMPLE: Table 5.1(b) shows the $\hookrightarrow$ edges introduced for our use case. We have three edges, representing the conditioning of $x_{M,S}$ to a low, medium, and high marketing budget, respectively (recall that $x_{M,S}.dim_1$ is modeled by x_M, see Table 5.1(a)).

The operators σ^{τ} and $\mathcal{A}_{MIN/MAX}$ require that predicates ϕ_i are captured for each $x_i \in X$. Each ϕ_i references intermediate processing information captured when the operator is initially executed[3].

For $\sigma^{\tau}(X, \omega(<, \upsilon))$, we record the selection probability p_i for each x_i. Assume that $p_i > \tau$, i.e., x_i fulfilled $\omega(<, \upsilon)$ with sufficient probability and was therefore selected into Y. Then, similar to [KLD11], an item x_i' deviating from x_i (with $p_i > \tau$) will become unselected only if $\Delta_{x_i}(\upsilon) < \tau - p_i$. Similarly for an unselected item ($p_i < \tau$), the item will get selected only if $\Delta_{x_i}(\upsilon) > \tau - p_i$. The procedure for predicates $\omega(>, \upsilon)$ is similar, and we omit it here.

In the procedure of $\mathcal{A}_{MAX}(X)$, the high percentile values $\dot{x}_{q,i} = inv(x_i, 0.99)$ computed for each $x_i \in X$ are captured. Further, we keep a tree structure $kTree$, in which we cache the k highest values $\dot{x}_{q,i}$ with a reference to their associated $x_i \in X$. For each $x_i \in kTree$, we then capture a predicate ϕ_i that evaluates to *true* if $\Delta_{x_i}(\dot{x}_{q,i}) \neq 0$. In this case, the new quantile value $\dot{x}'_{q,i}$ will deviate from $\dot{x}_{q,i}$ and thus must be recomputed. For all other items $x_i \notin kTree$ we define ϕ_i to be true if the deviation $\Delta_{x_i}(\dot{x}_{q,i})$ is *negative*, since only then $\dot{x}'_{q,i} > \dot{x}_{q,i}$ and x_i' may contribute to the k top values. The procedure for $\mathcal{A}_{MIN}$ is similar and therefore omitted here.

After this section discussed the rationale of the captured operator-specific edge and predicate information, the next section describes how those information are exploited for efficient result recomputation. As an alternative to exact recomputations based on known deviations,

[3]We list the intermediate processing information in the *out*-column of σ^{τ} and $\mathcal{A}_{MIN/MAX}$ of Table 5.2, even though they do not constitute the actual output, but rather provenance augmentations.

o_i	in	out	$\approx$ edges	$\hookrightarrow$ edges	Impact predicate ϕ
Univariate					
$\mathcal{T}$	$x, \omega : (\upsilon_1, \upsilon_2)$	$\dot{y}_p$	—	$d_{in} \longrightarrow d_p$	$\phi : (\Delta_x(\upsilon_1) \neq 0 \vee \Delta_x(\upsilon_2) \neq 0)$
$\mathcal{Q}$	x, p	$\dot{y}_q$	—	$d_{in} \longrightarrow d_q$	$\phi : (\Delta(x_q) \neq 0)$
σ^τ	$X, \omega : (\lesseqgtr, \upsilon), \tau$	$Y, (p_i)$		$d_{in} \longrightarrow d_{out}$	$\phi_i : (\Delta_{x_i}(\upsilon) \lesseqgtr \tau - p_i)$, if $x_i \in Y$
					$\phi_i : (\Delta_{x_i}(\upsilon) \gtreqless \tau - p_i)$, if $x_i \notin Y$
$\mathcal{A}_{SUM}$	X	$\dot{y}_{sum}$	—	—	—
$\mathcal{A}_{MIN}$	X	$\dot{y}_{min}$	—	$d_{in} \longrightarrow d_{min}$	$\phi_i : (\Delta_{x_i}(\dot{x}_{q,i}) \neq 0)$, if $x_i \in kTree$
		$(\dot{x}_{q,i})$			$\phi_i : (\Delta_{x_i}(\dot{x}_{q,i}) > 0)$, if $x_i \notin kTree$
$\mathcal{A}_{MAX}$	X	$\dot{y}_{max}$	—	$d_{in} \longrightarrow d_{max}$	$\phi_i : (\Delta_{x_i}(\dot{x}_{q,i}) \neq 0)$, if $x_i \in kTree$
		$(\dot{x}_{q,i})$			$\phi_i : (\Delta_{x_i}(\dot{x}_{q,i}) < 0)$, if $x_i \notin kTree$
$\mathcal{D}$	$\dot{X}, \beta$	y		—	—
Multivariate					
$\mathcal{C}$	x_j, x_C	y_{jnt}	$d_j \longrightarrow_{dim_j} d_{out}$,	—	—
	$j = 1, ..., m$		$d_C \longrightarrow_{cor} d_{out}$	—	—
$\mathcal{M}$	$x, \omega_j : (\upsilon_{j1}, \upsilon_{j2})$	y_{mrg}	$d_{in\ dim_k} \longrightarrow_{dim_k} d_{out}$	$d_{in\ dim_j} \longrightarrow d_{out}$	$\phi_{dim_j} : (\Delta_{x.dim_j}(\upsilon) \neq 0)$,
	$j = 1, ..., \lvert\Omega\rvert$		$k = \lvert\Omega\rvert + 1, ..., m$		$\upsilon_{j1} < \upsilon < \upsilon_{j2}$
$\mathcal{T}^m$	$x, \omega_j : (\upsilon_{j1}, \upsilon_{j2})$	$\dot{y}_p$	—	$d_{in} \longrightarrow d_p$	$\phi_{dim_j} : (\Delta_{x.dim_j}(\upsilon_{j1}) \neq 0$
					$\vee \Delta_{x.dim_j}(\upsilon_{j2}) \neq 0)$

Table 5.2: $\approx$- and $\hookrightarrow$-edges and impact predicates ϕ

the approximate evaluation of *assumed* changes in input data is addressed in the second part of Section 5.3.

5.3 Change Impact Analysis

As basis for evaluating change impacts through the computation process, we use information about the deviation of base data, Δ_x. Then the goal is to evaluate their impact on derived data.

5.3.1 Retrieving Derivation Information

The capture and querying of data derivation information, while not being the focus of this work, builds the basis for users (i) to find base data and operators applied in the derivation of a relevant result data and (ii) to analyze change impacts from base data on dependent data through the graph. As the *derivation* of d, we denote the part of a computation that finally yielded d, i.e., the sub-graph $P(d)$ of $\mathcal{G}$, $P(d) = P_O(d) \cup P_D(d) \cup P_E(d)$, where $P_D(d)$ are the contained data nodes, $P_O(d)$ the applied operator nodes, and $P_E(d)$ contains the edges connecting the nodes in $P_D(d) \cup P_O(d)$. We traverse $\mathcal{G}$ along its edges E to answer basic queries about the source and transformation provenance of data of interest. For a node d, all elements in $P_D(d)$ can be retrieved as the transitive closure $wdf^*(d)$. Similarly, we can access provenance for a specific version of a node d through $P(d, v_i)$, and traversing associated edges (either inherited from version v_{i-1} or annotated with v_i).

5.3.2 Recomputation

Consider a result data item x_t at node d_t and a deviation Δ_{x_b}, where $d_b \in P_D(d_t)$. Given $\mathcal{G}$, we can re-execute the computation that yielded x_t from scratch. In particular, we can reconstruct the query that yielded x_t by traversing $\mathcal{G}$ along wgb and $used$ edges starting at d_t to retrieve applied operators o_i, and their input and output data ($o_i.in = \bigcup d_j \mid \exists\ o_i \xrightarrow{used} d_j$ and $o_i.out = \bigcup d_j \mid \exists\ d_j \xrightarrow{wgb} o_i$). Then, we can map each operator node $o_i \in P_O(d)$ to its implementation and parameterize it using $o_i.par$. Finally, given all base data is available, we can recompute x_t.

At this point, we naturally want to exploit the captured operator-specific information and intermediate initial computation results to lower the costs for recomputations. For two versions of base data items, x, x' or datasets X, X', there can be different degrees of modification (f), which will influence the costs of recomputation. Particularly, for input data sets X, we denote the fraction of items $|X_\Delta|/|X|$ for which deltas are considered in the recomputation as f_X. To ensure an efficient recomputation, the overhead of the recomputation times in proportion to f_X should be small, e.g., if $f_X = 0.5$, the recomputation times should, ideally, be close to one half of the initial computation times. For individual items x, we denote the modified fraction of their support interval I_x as f_x. Again, with smaller values of f_x, the times for recomputation should decrease, due to the smaller amount of data loaded and processed. Section 7.6 will investigate the recomputation methods presented below regarding the achieved runtime efficiency for different f_X and f_x.

First, we consider the case of individual processing steps represented through a node o and its associated inputs. As described above, the goal is to exploit recorded information to *restrict* recomputation steps to incorporate only *impact-relevant* deviations Δ_x, i.e., those Δ_x for which ϕ evaluates to *true*. In the second part, the optimized recomputation of operators over multivariate data is considered, where information reflected through $\approx$- and $\hookrightarrow$- edges is used for a more efficient processing of operator sequences.

Recomputing the basic statistic operator $\mathcal{T}(x, \omega)$ based on deviations Δ_x^H amounts to testing $\phi(\Delta_x^H(v) \neq 0)$ and computing $\dot{x}'_{out} = \dot{y} + \Delta_x^H(v)$. This is equal to computing $\mathcal{T}$ from scratch when Δ_x^H affects the whole interval of x. For $\mathcal{Q}(x, p)$, the predicate $\phi(\Delta_x^H(\dot{x}_q) \neq 0)$ is tested, and $\dot{x}'_{out} = \mathcal{Q}(x, p)$ is then recomputed only in cases where ϕ evaluates to *true*. This induces a complexity of $\mathcal{O}(log_2\beta)$, which is reduced to $\mathcal{O}(1)$ if ϕ evaluates to *false*.

Probabilistic Threshold Selection

Given a data set $Y = \sigma^\tau(X, \omega)$, we want to compute $Y' = \sigma^\tau(X', \omega)$ based on the initial result X and deviations of data items $x'_i \in X'$. As before, the assumption is that for each x_i, the corresponding (i.e., selected) item in Y is accessible based on a unique key. Then, we start with a result set $Y' = Y$, and remove each item for which ϕ_i evaluates to *true*. Similarly, we add each (previously unselected) item $x'_i \in X' \setminus Y$, to Y' if $\phi_i = true$, and update the selection probabilities p'_i accordingly.

Aggregation

To recompute $\mathcal{A}_{SUM}$ based on X', we incorporate each deviation $\Delta_{x_i}^H$ by computing an *expected delta value*

$$\mathbb{E}^\Delta(\Delta_{x_i}^H) = \sum_{j=0}^{\beta} \Delta_{x_i}^H(v_j) \cdot v_j$$

at β equi-distant points v_j in the interval I_{x_i}. Then, we recompute $y' = y + \sum_i \mathbb{E}^\Delta(\Delta_{x_i}^H)$[4].

For $\mathcal{A}_{MAX}(X)$, we consider a one-time recomputation based on a deviating set of input values, X'. Recall that, at the time of recomputation, we have available the quantile values $\dot{x}_{q,i}$ and predicates ϕ_i which are recorded when $\mathcal{A}_{MAX}(X)$ is initially computed. First, for any ϕ_i of an item $x_i \in kTree$ that evaluates to *true*, we compute $\dot{x}'_{q,i} = inv_{x'_i}(0.99)$ as the new percentile value of x'_i and update $kTree$ with the new value. Similarly, for all $x_i \notin kTree$ we compute new quantiles $\dot{x}'_{q,i}$ if ϕ_i evaluates to true (since only then, $\dot{x}'_{q,i} > \dot{x}_{q,i}$) *or* if the old value $\dot{x}_{q,i}$ now is larger than the current highest value in $kTree$. We successively update $kTree$ with the newly computed $\dot{x}'_{q,i}$ and return the new absolute maximum value $\dot{y}'_{max}$. The procedure for $\mathcal{A}_{MIN}$ is similar and therefore omitted.

[4]Note that we write $\mathbb{E}^\Delta(\Delta_x^H)$ even though Δ_x^H is no distribution function but rather reflects the deviation between two distributions; yet the computation of the expected delta value $\mathbb{E}^\Delta$ is similar to that of an expected value of a distribution, $\mathbb{E}$.

Multivariate Data Analysis

In the standard approach for multivariate analysis, we either have available a joint distribution x_{jnt} or first compute it using $\mathcal{C}$, and then process it using $\mathcal{T}^m$ or $\mathcal{M}$. A deviation in any of the components of x_{jnt} (dim_j and cor) will affect x_{jnt}. Therefore, considering edges $d_j \overset{\approx}{\longrightarrow}_{dim_j} d_{jnt}$, we need to compute a new data item x'_{jnt} if there is a deviation in any of the input marginals x_j. Then, we need to consecutively re-evaluate operators that *used* x_{jnt} as input. For example, for an edge $o_{\mathcal{M}} \overset{used}{\longrightarrow} d_{jnt}$, we recompute $\mathcal{M}$ over x'_{jnt}. Using the operator-per-operator approach, this recomputation must be done even if no deviation Δ_{x_j} (at the input of $\mathcal{C}$) will ultimately impact the result of $\mathcal{M}(x'_{jnt}, \Omega)$. Moreover, the complete distribution x'_{jnt} needs to be recomputed, even when only a restricted part of x'_{jnt} (satisfying all $\omega_j \in \Omega$) eventually contributes to the result of $\mathcal{M}$. The same holds true for the computation of $\mathcal{T}^m$. Using the recorded $\approx$ and $\hookrightarrow$ edges, analyses over multivariate data can be processed more efficiently as described in the following.

Restriction push-down One can exploit knowledge about impact restrictions introduced by an operator o_i by pushing them down to the input of a direct preceding operator o_{i-1}. Consider the edges $d_j \overset{\approx}{\longrightarrow}_{dim_j} d_{jnt}, j = 1, ..., m$ (which is introduced as $\mathcal{C}$ is applied) and $d_{jnt}\,{}_{dim_j}\!\overset{\hookrightarrow}{\longrightarrow} d_{out}$ (introduced as $\mathcal{M}$ is applied to marginalize out $dim_j, j = 1, ..., |\Omega| < m$). Now, it is possible to infer implicit $\hookrightarrow$ edges $d_j \overset{\hookrightarrow}{\longrightarrow} d_{out}$. Conceptually, the impact-restriction introduced by $o_{\mathcal{M}}$ is thus pushed down to the input node of $o_{\mathcal{C}}$, d_j. On this basis, both $\mathcal{C}$ *and* $\mathcal{M}$ need to be recomputed only if any deviation Δ_{x_j} will impact the output of $\mathcal{M}$ according to the impact predicates associated with $\mathcal{M}$. The approach applies similarly to the recomputation of $\mathcal{T}^m$.

Section 7.6 evaluates how this impact-based restriction decreases the overhead of recomputations for different degrees of f_X, where f_X denotes the fraction of the *pairs of marginals* x_j that satisfy ϕ_j.

Rewriting based on $\approx$-correspondences Second, correspondences between distributions components can be exploited in order to re-arrange the operator plan associated with an analysis. Here, the goal is to *restrict the recomputation* to the relevant part of x_{jnt} (i.e., the part that will ultimately influence the result of $\mathcal{M}$ and $\mathcal{T}^m$). This approach has been described as optimization possibility in Section 4.5.1. Here, it is briefly discussed how the recorded $\approx$-relations enable this optimization. In particular, the rewriting is based on exploiting (a) knowledge about corresponding dim_j components of x_C and x_{jnt} and (b) the mapping between distributions x_C and x_{jnt} (using functions $inv_j : x_C.dim_j \rightarrow x_{jnt}.dim_j$ and $cdf_j : x_{jnt}.dim_j \rightarrow x_C.dim_j$) as discussed in Section 4.2.

Consider a bivariate case with x_1, x_2 as marginals, and predicates $\omega_1(v_{11}, v_{12})$ and $\omega_2(v_{21}, v_{22})$ used in the computation of $\mathcal{M}$ and $\mathcal{T}^m$. Then, consider a recomputation based on inputs x'_1 and x'_2. For $\mathcal{M}$, instead of computing $\mathcal{M}(x'_{jnt}, \omega_1)$ with $x'_{jnt} = \mathcal{C}(x'_1, x'_2, x_C)$, we compute only the relevant region of x'_{jnt} as input to $\mathcal{M}$. That is, we compute $\mathcal{M}(x'_{jnt}, \omega_1)$ with $x'_{jnt} = \mathcal{C}(x'_1, x'_2, x'_C)$, where x'_C is *floored* outside the region $[cdf_{x'_1}(v_{11}), cdf_{x'_1}(v_{12})]$. For $\mathcal{T}^m$ we can directly compute the joint probability over x_C exploiting knowledge about the corresponding dim-components. Therefore, instead of computing $\mathcal{T}^m(\mathcal{C}(x'_1, x'_2, x_C), (\omega_1, \omega_2))$, the computation can be rewritten to $\mathcal{T}^m(x_C, (\omega_{C1}, \omega_{C2}))$ with predicates $\omega_{C1}(cdf_{x'_1}(v_{11}), cdf_{x'_1}(v_{12}))$ and $\omega_{C2}(cdf_{x'_2}(v_{21}), cdf_{x'_2}(v_{22}))$.

Having discussed different approaches to exploit captured provenance information for a recomputation using Δ^H, the next section addresses an approach for approximate recomputation, where assumed deviations are represented through a function representation.

5.3.3 Approximate Deviation Analysis

Often, a user may either not know accurate deviation information or require only an approximate recomputation of results. This is the case when he wants to evaluate the effect of an *assumed* deviation (rather than evaluating a known or expected plan-actual deviation). For example, in our use case the user may wish to "shift" distributions associated with per-customer sales (X_S) such as to investigate the effect of a skew in $x_{S,i}$ towards their lower tails. In this case, he can use an approximate representation of deviations.

Delta Function Approximation Δ^T_x For an exemplary approximate representation, we apply a simple piecewise linear function Δ^T_x. As the deviation of two CDFs by definition approaches zero as we draw near $l^{x'}_x = min(\underline{x}', \underline{x})$ and $h^{x'}_x = max(\overline{x}', \overline{x})$, we use a triangular function defined in $[l^{x'}_x, h^{x'}_x]$. This is adequate in cases where deviations between x, x' are gradually increasing or decreasing, rather than occurring at one position of the distribution only. The parameters of Δ^T are the (assumed) amount of the maximum distance between $cdf_{x'}$ and cdf_x, $\bar{\delta}^{x'}_x$, $0 \leq \bar{\delta}^{x'}_x \leq 1$, and the location of that deviation, $pos^{x'}_x$. We define

$$\Delta^T_x(v) = \begin{cases} \bar{\delta}^{x'}_x \cdot (h^{x'}_x - v)/(h^{x'}_x - pos^{x'}_x) & \text{if } v \in [pos^{x'}_x, h^{x'}_x], \\ \bar{\delta}^{x'}_x \cdot (v - l^{x'}_x)/(pos^{x'}_x - l^{x'}_x) & \text{if } v \in [l^{x'}_x, pos^{x'}_x], \\ 0 & \text{otherwise.} \end{cases}$$

While the Δ^H representation of deviations requires storage, access and processing costs equal to x in the worst case (i.e., if the deviation affects the complete interval I_x), Δ^T is entirely represented by four parameters. Those are stored in a function table from which the delta associated with an item x' is accessed at query time.

Approximate Recomputation Using Δ^T, users can now also reevaluate queries by incorporating the approximate deviation information reflected by Δ^T. Since Δ^T retains (approximate) information about the location of the distance between distributions, one can use it similarly to Δ^H, i.e., to compute the deviation Δ^T_x at a given position v, such as to use the result to evaluate ϕ in the process of operator recomputation. At the same time, using Δ^T we can avoid the higher storage and access costs incurred by Δ^H.

Naturally, in cases where Δ^T is used as an approximation of Δ^H (rather than to introduce an assumed deviation), there will be an approximation error introduced by Δ^T which influences the accuracy of the approximate recomputation. In particular, if the triangular shape (or any other function approximation one may devise) of Δ^T does poorly or not at all fit the real deviation, the recomputed result will naturally be inaccurate as well. The application of more complex and accurate approximation functions is out of the scope of this thesis, but could be easily integrated with the discussed approaches.

5.4 Summary

This chapter discussed the capture of provenance information for scenario analysis processes, and its exploitation with the goal to efficiently re-evaluate analytic computations over continuously distributed data. The issue of recomputations is motivated with respect to incorporating plan-actual deviations in data, as well as investigating assumed input deviations for change impact analyses. The presented approaches focus on analyses that involve the application of the analytic operators discussed in Sections 3 and 4. Those operators share the common characteristic that their results often rely on a restricted region of their input data distributions only. According to the presented approaches, this information is captured in the form of provenance relations and predicates, where the former are recorded on the level of operator applications (and their input and output data sets), while the latter are stored per data item. Additionally, intermediate processing information, such as the selection probability values in the case of the operator $\sigma^{\mathcal{T}}$, are kept in memory. At re-evaluation time, the captured information is exploited to perform the required computation steps only selectively. That is, the recorded predicates are evaluated in order to determine impact-relevant changes in input data. Then, further computation steps are only re-computed for those data that exhibit impact-relevant changes.

In Section 7.6, the efficiency of the presented techniques for recomputation of analytical queries will be evaluated. Noteworthy, in this thesis, the focus is mainly on incorporating *manual* changes to input data as well as possible plan-actual deviations that occur when the distributions of actual data deviate from originally assumed deviations. Another important and promising application of the recomputation approach is the *automatic* computation of deviating scenarios. When the scenario space shall be explored for (combinations of) deviations in input data to identify critical or promising neighboring scenarios, this results in a large number of possible deviations. In this case, the efficiency of subsequent scenario computations under partial changes in the input data becomes even more apparent. This topic, which is similar to the optimized computation of what-if queries under varying parameterizations investigated in [KN11], is highly interesting yet out of scope of this thesis.

Although the generic capture and handling of provenance for arbitrary queries is not addressed in this thesis, it is noteworthy that the presented approach can be extended to allow for the representation of dependency (provenance) in a more generic manner. Particularly, predicates applied in the processing of operators such as $\mathcal{T}^m$ and $\mathcal{M}$ can be reflected by capturing *conditions* for the tuples associated with individual data items in a dedicated condition column, in line with the representation system of probabilistic c-tables (see Section 6.4.5).

Having presented the concepts of this thesis during the last three chapters, the next chapter is dedicated to the presentation of related work in the area of uncertain data management and scenario analysis. There, we will present important (alternative) techniques for representing and processing uncertain data, dependency and provenance, as well as a discussion of a selected range of existing solutions.

6 Related Work

This chapter discusses related work in the areas of uncertain data management and provenance. First, prominent alternative *technologies* for modeling and processing uncertainty and provenance in data are discussed. This includes foremost the *implicit approach* to uncertain data management using *Monte Carlo* simulation, the explicit representation of probabilistic data with dependencies through *probabilistic graphical models* and *c-tables*, and the management of provenance in its different forms. Then we consider a number of existing systems for probabilistic data management. We discuss their relation to the context of scenario analysis and planning, and how their capabilities differ from the approaches presented in the thesis at hand.

6.1 Monte Carlo Simulation

The approaches discussed in this thesis rely on an explicit representation and processing of uncertainty to the largest part. A notable exception is the concept of sampling copulas and their application for correlation handling discussed in Chapter 4. As stated before, the class of approaches that model uncertainty implicitly, relying on Monte Carlo (MC) simulation, form a widely-used alternative to the explicit representation of uncertainty.

6.1.1 The Monte Carlo Approach

The essence of the MC simulation approach is the simulation of a large number of stochastic events which are then evaluated deterministically. The MC approach bas been widely applied in many fields such as physics, engineering, finance, telecommunications, and games. Generally, the goal of MC is to compute a numerical solution to problems (functions) which would be expensive or even impossible to solve analytically. To this end, one first draws a large number of samples for each of the input variables of the problem (the function to be evaluated) through appropriate sampling functions. The generated samples are then considered realizations of the problem in different possible worlds, and the function of interest is evaluated deterministically over the samples. This way, it is possible to evaluate arbitrarily complex functions over (multivariate) input distributions, as well as to compute statistical measures over the function outputs, such as their expected value.

The great benefit provided by the MC simulation approach, as opposed to the explicit representation of uncertainty with its hard-wired uncertainty representation, is its genericity. The representation of uncertainty is encapsulated completely in the sampling functions. Therefore, the class of applicable probability representations is not restricted. On the one hand, this implies great flexibility since the user can model any distribution through appropriate sampling functions when required. On the other hand, this genericity can come at the cost of potentially large memory and processing costs, especially as the complexity of the applied statistic models increases. Apart from this aspect, models involving very complex sampling functions can be hardly accessible or comprehensible to users without a deep knowledge in statistics. Thus, they may be hard to maintain, adjust, and reuse.

When using MC in a *sample first approach* (as described for a specific system in Section 6.4.4), all random variables involved in a query are represented symbolically and their concrete instantiation is realized through sampling at runtime. Then, queries are evaluated over the potentially very high number of all samples. As a variation to the pure sample-first approach, the relational parts of queries can be evaluated partially before the sampling phase is started. This can help to avoid wasting sampling effort on such samples which will be discarded during the further evaluation of the query (e.g., based on a selection predicate over a deterministic attribute). An implementation of such an approach is described in Section 6.4.5.

Numerical Integration One common application of MC is the numerical integration of functions for which an analytical evaluation is infeasible or even impossible. Given a distribution function P_x, one can draw a very high number of samples following the distribution, and then evaluate the integral $\int_{v_1}^{v_2} P_x(x)dx$ within a lower bound v_1 and an upper bound v_2 as the fraction of those samples that fall in the interval $[v_1, v_2]$.

While the approaches of this thesis mostly rely on an explicit representation of probabilistic data, sampling-based numerical integration is used in the context of correlation handling. More specifically, we integrate over sampled multidimensional copula functions in the course of constructing ACRs, as well as in the process of evaluating joint distributions based on samples drawn from a copula function, or based on an ACR, respectively.

6.1.2 Optimizations

There exist a variety of MC algorithms that are tailored to the specific nature of an analysis or optimized regarding the accuracy of simulation results. A primary concern of those optimization methods is to focus the sampling effort on those regions that are most relevant for the evaluated function. An important application is the simulation of rare events and the evaluation of queries over such events, which is addressed, e.g., in the work of [JXW$^+$10].

Importance Sampling and Recursive Stratified Sampling As an example for an optimized sampling approach, the *importance sampling* technique is used to draw samples from a distribution of input variables in such a way that it “emphasizes” (i.e., draws more frequently) those values that have more impact on the function estimate than others. This way, the error of the final result estimate can be reduced with a lower total number of samples than would be required using the basic MC approach. The bias that is introduced to the sampled input distributions is accounted for by appropriately weighing the resulting outputs of each simulation iteration in the computation of the final result estimate.

Similarly to importance sampling, *recursive stratified sampling* approaches aim to increase the accuracy of the function estimate, while minimizing the number of required samples. This is achieved by focusing the sampling effort on those regions in the m-dimensional sample space where the variance of the function value is the largest. The algorithm proceeds recursively by first computing a function estimate and error estimate with input samples being obtained through the basic MC approach. Given the resulting error is larger than an accuracy threshold, the multidimensional space is divided into subregions and the first step is repeated for each subregion in turn. This procedure is repeated until the error estimate is low enough, and the simulation results are returned.

In this work, copula function samples are used for the representation of correlation structures. However, the proposed ACR approach does not perform sampling at runtime but rather uses sampling-based approximate representations of the copulas. Those are produced offline before they are imported to the database to be used for future queries. The accuracy and runtime costs of the operators for correlation processing depend solely on the representation accuracy of the applied ACRs. Due to the offline construction of the ACRs, the sampling costs are not considered of prior importance within this work. Indeed, we hold that the number of underlying samples can be potentially very large, so that the produced ACR reflects the copula function closely. Specific relevant regions of the copula (such as the joint tails) can then be represented highly accurately by choosing an appropriate nested histogram representation, as discussed in Section 4.3.4.

Markov Chain Monte Carlo (MCMC) The class of MCMC algorithms can be applied when a desired joint distribution is not known or is difficult (expensive) to sample from, but the individual *conditional distributions* of the variables are known. The overall method is to construct a Markov chain whose *stationary distribution* is then the desired sample distribution. This is achieved by performing a *random walk*, at each step of which a sample from each one-dimensional conditional distribution is generated in turn. The resulting sequence constitutes a Markov chain which converges to the stationary distribution after a certain number of steps.

Examples for MCMC techniques include the Metropolis-Hastings [Has70] algorithm and Gibbs sampling [GG84] as a special form thereof. They are particularly well-suited to derive the posterior distribution of a Bayesian network (which is essentially constructed as a graph of conditional distributions). Gibbs sampling is applied in the work of [JXW$^+$10] to efficiently sample from the *tails* of a query result distribution and thus speed up the evaluation of queries over light-tailed result distributions considerably. The method described in [WMM10] relies on Metropolis-Hastings. In this work, an uncertain database is modeled through a database that represents a single initial possible world, and an associated factor graph that encodes the distribution over possible worlds. The MCMC technique is applied to infer further possible worlds from the initial possible world, such as to “recover” the uncertainty in the data to a desired degree.

6.2 Explicit Representation of Probabilistic Data with Dependencies

Considering the representation of probabilistic data with dependencies, we address two models which received broad attention in the field of probabilistic data management. First, we consider the representation of joint distributions through a graph structure reflecting the dependencies between the involved variables. Such graphs form the basis for so-called *Probabilistic Graphical Models (PGMs)*, which are applied heavily for modeling probabilistic information in the fields of AI and knowledge management, e.g., for natural language processing, and information extraction tasks. In databases, the graphical modeling approach has been adopted in order to allow the representation of complex distributions over large sets of random variables and for efficient querying (inference) over such data. In [DGS09], the authors discuss the use of PGMs for uncertain data management. This section gives a brief account of the PGM approach, adopting the notation of their work. Afterwards, the concept of *probabilistic conditional tables* (probabilistic c-tables) is discussed as an approach that was developed primarily in the field of PDBs as a probabilistic data model that is closed and complete under the relational algebra.

6.2.1 Probabilistic Graphical Models

Probabilistic graphical models allow for an efficient representation and query processing over large volumes of uncertain data. In brief, they represent dependencies over a set $\mathcal{X} = \{X_1, ..., X_n\}$ of random variables, each of them associated with uncertain tuples or attribute values, through a graph structure and associated probability values. Assuming each of the variables X_i is *m-valued*, a basic approach would be to store m^n numbers (conditional probabilities) for the joint distribution between the n variables. This is an expensive approach for larger numbers of m, since it implies exponential growth in the representation of the joint distribution. However, as stated in [DGS09], real-world distributions often exhibit conditional independence between some of their variables. This aspect can be exploited when using PGMs.

A PGM represents the dependency structure of all variables in $\mathcal{X}$ through a graph. The graph contains nodes representing the individual variables in $\mathcal{X}$, and edges representing *direct* dependencies between those variables. If two nodes in the graph are *directly* connected, it means that they interact (i.e., depend on each other) directly. If two variables are, however, not *directly* connected, they are *conditionally independent*.

Factored Representation The conditional independence between subsets of variables can be reflected in a *factored representation* of the modeled joint distribution. Factors essentially represent the interaction within the sets of (dependent) variables in a graph; conditionally independent sets are associated with different factors. Therefore, the factored representation allows to represent a (complex) joint distribution over a set of variables $\mathcal{X}$ by means of multiple factors of independent subsets $\mathcal{X}_i \subseteq \mathcal{X}$.

Adopting the notation from [SDG09], a *factor* is defined as a function $f(\mathbf{X})$ over a set of variables $\mathbf{X} = \{X_1, ..., X_n\}$ such that $0 \leq f(\mathbf{X} = \mathbf{x}) \leq 1$. The complete joint distribution over all variables in $\mathcal{X}$ can then be computed as the product of factors,

$$Pr(\mathbf{X} = \mathbf{x}) = \prod_{i=1...m} f_i(\mathbf{X}_i = \mathbf{x}_i)\,.$$

Using factors, a graphical model $\mathcal{P} = (\mathcal{F}, \mathcal{X})$ can now represent a joint distribution over all variables $X_i \in \mathcal{X}$ with each of the factors $f_i \in \mathcal{F}$ being defined over a subset of $\mathcal{X}$.

Bayesian vs. Markov Networks Based on the structure of the graph representation, one distinguishes the classes of *Bayesian networks* (in case the graph is a directed acyclic graph) and *Markov networks* (in case the graph is undirected). The two classes differ both in the form of dependencies that can be expressed, as well as the factorization one can apply, as further detailed in [DGS09]. While the use of Bayesian networks is more suited and natural to model dependencies that are *causal* and asymmetric, applying Markov networks is more appropriate when there is no directionality of influence of one variable on another (or when no such directionality is known), i.e., the dependency structure is symmetric.

Inference over PGMs A query over uncertain data represented through a PGM is evaluated through inference over the graph representation. The structure of the graph and, particularly, its degree of factorization, can be exploited to increase the query efficiency.

Typical queries that are performed over a PGM include the computation of the *conditional probability* $Pr(\mathbf{Y}|\mathbf{E} = \mathbf{e})$, i.e., the probability distribution of a subset of variables $\mathbf{Y}$ under the condition of an assignment $\mathbf{E} = \mathbf{e}$. This includes the special case of computing a marginal distribution $Pr(\mathbf{Y})$, where all variables $\mathcal{X} \setminus \mathbf{Y}$ are marginalized out (see Section 2.4.1).

Using the PGM representation, one can exploit the particular characteristics of the graph structure for a more efficient computation of such queries, instead of applying the "naive" approach of computing queries for the complete joint distribution (which results in exponential complexity). Examples for highly efficient inference techniques include the Variable Elimination (VE) [DD96] and junction tree [CDLS03] algorithms.

PGM in PDBs The graphical modeling technique is applied by different solutions for probabilistic data management, such as [SDG09] (see Section 6.4.2) and [WMGH08, WMM10]. In [WMM10] a graphical model is used to represent the joint distribution over the possible worlds of a PDB, which serves as a basis for creating a desired number of possible realizations of the database from an initial world in the process of query evaluation.

Existing work that considers the use of PGMs in databases focuses on modeling and processing discrete joint distributions. However, the concept of copulas and the use of ACRs, which was investigated in Chapter 4, could be integrated with the graphical model approach, so as to enable an efficient processing of complex joint distributions with continuous marginal distributions. Conceptually, copulas (and ACRs as their approximate representations) can then be considered as factors $f_i \in \mathcal{F}$ of the graphical model, and conditional independence between subsets of variables can be exploited as described above. Such an approach would serve to mitigate the high runtime and memory costs that occur in the case of larger-dimensional copulas (ACRs), which has been addressed as a restriction of the ACR approach in Section 4.3.3. More specifically, using a factored representation of an m-dimensional copula (ACR), the representation and processing of the m-dimensional joint distribution can be split up into several factors, each represented through a lower-dimensional ACR. Then, similar to the representation of (conditional) dependency structures between discrete variables, one can represent (conditional) dependencies between continuously distributed variables. This application of copulas is close to the approach of the so-called Copula Bayesian Models discussed in [Eli10], which have been developed in the context of machine learning.

6.2.2 Probabilistic C-Tables

The use of *probabilistic c-tables* as a representation system for incomplete, probabilistic data with dependencies was considered in the work of [GT06]. Probabilistic c-tables are based on the concept of c-tables [IL84]. In a c-table, the attribute values of each tuple can take either a constant value or a variable, thus enabling the representation of incomplete information. In addition to standard data columns, each tuple of the relation can be associated with a *local condition*. A local condition combines atomic conditions over variables and constants through conjunction, disjunction, and negation. Given a specific variable assignment (i.e., mapping each variable in the table to a value from its domain), the condition can either evaluate to true (i.e., be satisfied under a given variable assignment) or false. This way, c-tables enable the representation of dependency between variables. A possible world over c-tables is specified by a particular variable assignment. A relation in such a possible world is then obtained from each c-table by dropping all tuples whose conditions are not satisfied under the given assignment. C-tables are closed under relational algebra. Conceptually, the local conditions associated with tuples of a c-table can be considered equivalent to the concept of data provenance in that they specify the existence of a tuple based on the evaluation of atomic conditions over other variables.

A probabilistic c-table is a c-table in which each variable is defined as a *finite random variable*. That is, besides the conditions columns one also stores information about the probability distributions of the values taken by the variables contained in the table. Vertical partitioning can be applied to represent attribute-level uncertainty. That is, tuples in a table can be decomposed in order to efficiently represent several *independent* uncertain attributes. Probabilistic c-tables are complete and closed under the relational algebra.

Probabilistic c-tables have the same expressiveness as PGMs. Both representation systems can essentially be used to model any joint distribution between a number of random variables, and thus, any joint distribution that can be produced using the correlation introduction operation discussed in Chapter 4. The vital distinction between the concept of ACRs and the PGM and c-table-based representation systems is that ACRs indeed represent the correlation information between *generic marginal distributions*, while PGMs and probabilistic c-tables represent *specific* joint distributions.

6.3 Data Provenance

There exists a variety of approaches to capture and process *data provenance* in a relational setting. A first formal classification in [BKT01] distinguished existing approaches into so-called *why-provenance* (describing why an output data item was constructed from input data), *how-provenance* (capturing how exactly output items have been constructed from input items), and *where-provenance* (describing where some output data was derived from in the input data). This section briefly discusses the notions of why-, how-, and where-provenance and describes a framework that provides a generic formalization to those different approaches.

6.3.1 Why, How-, and Where-Provenance

As stated before in Section 2.5, data provenance refers to different forms of information that capture the relation between the output data and input data of queries. To this end, one generally assumes that the items in the input data are annotated, such that the provenance of an output can be represented by means of references to the relevant input items. Those annotations can be at different levels (annotating tables, tuples, attributes, or individual attribute values) depending on the form of provenance information. To illustrate different approaches to data provenance, consider the input relations S and P shown in Table 6.1 and the relation R computed through the query $q(S, P) = \Pi_{name}(S \bowtie_{S.pid=P.pid} P)$.

(a) Sales S

sid	pid	
$s1$	$p1$	r
$s2$	$p1$	s

(b) Products P

pid	name	
$p1$	$name1$	u
$p2$	$name2$	v

(c) Sold R

name	
$name1$	t

Table 6.1: Input relations S, P, R and query result $R = q(S, P)$

Why-Provenance One of the first formal descriptions of provenance is presented in [CWW00], where the concept of *lineage* is introduced. There, the *lineage* of a tuple t in the output of a query comprises all tuples in the input data that in some way contributed to the production of t. For the tuple t in R, this includes tuples r, s, and u. A formally more precise notion of provenance was given in [BKT01] under the term of *why-provenance*. Why-provenance considers, for a given result tuple t, the set of so-called *witnesses* that contributed to the result tuple, rather than considering all input tuples that can contribute to t. Thus, a *witness* specifies a set of tuples which is in itself *sufficient* to produce output t; several such witnesses may exist for a single output tuple. For example, for the result tuple t in R, we have two witnesses, $\{r, u\}$ and $\{s, u\}$, each of which can individually produce t.

How-Provenance In addition to providing information about which tuples (or witnesses as sets of tuples) contributed to a query result, the notion of *how-provenance* further considers the question *how* individual tuples contributed to produce an output tuple. For example, for the tuple t in R, u has contributed in different ways, namely, by joining with several other tuples (r and s). This more informative (relative to why-provenance) level of provenance can be captured through so-called provenance polynomials, which have been formalized in terms of *provenance semirings* [GKT07] (see next section).

Where-Provenance While why-provenance describes the tuple combinations that serve as witnesses of an output tuple t, where-provenance describes exactly where a specific data *value* in the output (such as the value of the *name* attribute of tuple t) was "copied" from. That is, given the witness $\{r, u\}$ for the output tuple t with attribute $t.name$, the where-provenance returns the exact location (i.e., the attribute or field) in tuple u, from which the value of $t.name$ was copied from according to the executed query, i.e., the field $u.name$.

6.3.2 Provenance Semirings

As described above, different sorts of provenance information may be relevant depending on the application context. The classes of why-, how-, and where-provenance each address the capture of provenance in different contexts and on different levels of data. The genericity of the task of capturing and querying provenance information has been addressed in the work on provenance semirings of [GKT07]. There, the authors show how a *commutative semiring structure* can serve to model the source provenance information for results of positive relational (SPJU) queries in a generic manner. Provenance semirings can be applied to capture provenance information on many levels of detail and can be specialized to different application contexts, such as information integration, access control, and uncertain data management. The remainder of the section gives a brief account of the concept of semirings and their specialization to different implementations of provenance.

The Commutative Semiring Structure

A semiring is an algebraic structure consisting of a domain and two binary operations with their neutral element. For a semiring to be commutative, both operations must be commutative. As shown by [GKT07] polynomials with coefficients from the set of natural numbers $\mathbb{N}[X]$ serve as the domain most informative for capturing provenance annotations, reflecting the notion of *how-provenance* in [BKT01]. The corresponding semiring, with the commutative operations $+$ and $\cdot$ and the neutral elements 0 and 1 is denoted by

$$(\mathbb{N}[X], +, \cdot, 0, 1) \tag{6.1}$$

The neutral elements 0 and 1 are used to denote that an element is contained or is not contained in a relation, respectively. The variables of a provenance polynomial resulting from a query are the annotations of the input data items of this query. The operations $+$ and $\cdot$ are associated with the operators of positive relational algebra. In particular, the operation $+$ is associated with the application of union and projection, combining annotations of input tuples into the annotation of the tuple that results from the union or projection. The operation $\cdot$ is associated with the application of joins, combining the annotations of joined tuples. The coefficients and exponents of a provenance polynomial then reflect information about the number of possible ways the annotated tuple can be produced. To illustrate the use of semirings, consider the example from [GKT07] with a relation R (see Table 6.2(a)) and the SPJU query $q(R) = \pi_{AC}(\pi_{AB}R \bowtie \pi_{BC}R \cup \pi_{AC}R \bowtie \pi_{BC}R)$. In the relation R, we can see the annotations $p, r,$ and s for the three contained tuples.

(a) R

A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

(b) $q(R)$

A	C	Provenance
a	c	$2p^2$
a	e	pr
d	c	pr
d	e	$2r^2 + rs$
f	e	$2s^2 + rs$

Table 6.2: Input relation R and query result $q(R)$ with provenance polynomials

Looking at the provenance polynomials associated with the result $q(R)$ displayed in Table 6.2(b), we can see that the tuples annotated with p and r contributed to the creation of the result tuples (a, e) and (d, c). Also, we can see that tuple (d, e) can be produced in three ways, where two involve the tuple annotated with r twice (leading to $2r^2$) and one involves both the tuples annotated with r and s. This general representation of provenance information and their processing can now be specialized to particular applications by, first, discarding information and, second, applying appropriate operations with their associated neutral elements.

Specialization

The specialization of (provenance) semirings relies on the operation-preserving characteristics of *homomorphisms* (see, e.g., [Beu03, GKT07]). The generic algebraic structure of commutative semirings can be mapped to specific (less general) ones, which can then be used to appropriately handle provenance for particular (less general) application cases. This way, one can flexibly choose the most efficient provenance representation for a task at hand by *leaving out* information present in the provenance polynomials of the most generic semiring $((\mathbb{N}[X], +, \cdot, 0, 1))$. In principle, one can first carry out all computations within the general structure (that would be $\mathbb{N}[X]$) and map the results to a specialized structure in a last step. Alternatively, one can map to the specialized (less informative) structure first and carry out the computations on this representation afterwards.

For example, the semiring can be specialized to the model of "lineage" that is applied in [ABS+06]. Conceptually, this is achieved by discarding information about the number of times a tuple participated in one derivation of a query result, i.e., dropping the exponents from the provenance polynomial. The resulting algebraic structure is $(Trio(X), +, \cdot, 0, 1)$; it is termed Trio-lineage in [GKT07] since this form of provenance has been implemented in the Trio database (see Section 6.4.1). The changes in the query results from Table 6.2 are shown in Table 6.3(a). Note that one can no longer deduce that the tuple annotated p contributed twice to produce the result tuple (a, c).

For further specializing the semiring in Equation 6.1, one can drop both the exponents

(a) $K = Trio(X)$

A	C	provenance
a	c	$2p$
a	e	pr
d	c	pr
d	e	$2r + rs$
f	e	$2s + rs$

(b) $K = Why(X)$

A	C	provenance
a	c	$\{p\}$
a	e	$\{p, r\}$
d	c	$\{p, r\}$
d	e	$\{r, s\}$
f	e	$\{r, s\}$

Table 6.3: Query answers for $K = \text{Trio}(X)$ and $K = \text{Why}(X)$

and the coefficients of the polynomials. This results in the concept of why-provenance described above. The associated algebraic structure is $(Why(X), \cup, \cup, \emptyset, \{\emptyset\})$. The corresponding query results for why-provenance are presented in Table 6.3(b). We can see that the provenance information no longer reflects how or how often an input tuple contributed to the derivation of a result tuple.

The discussed provenance semirings offer a very generic approach to provenance capture for positive relational algebra (SPJU) queries. They also enable an efficient update (and deletion) propagation for such queries. For example, the deletion of a tuple can be represented by setting its annotation to the value 0, and propagating the change through the provenance polynomials that include the same annotation. In the context of probabilistic data, one can similarly reflect and propagate changes in the confidence values associated with specific tuples or attribute values. This has been exploited, e.g., in the work on sensitivity and explanation analysis for positive relational queries over discrete probabilistic data in [KLD11]. In the continuous case, however, modifications in input data affect a specific *part* of the distribution, rather than confidences associated with a discrete value. The impact on the resulting output of operations must therefore be determined based on the whole domain of the input distribution, as discussed in Chapter 5. The provenance polynomials can, furthermore, not reflect the application of analysis operations beyond the class of relational operators, such as those applied in this thesis. However, for the purposes of scenario provenance capture and recomputation, one requires such information. In the context of this thesis, the scenario derivation information is therefore captured on the level of operator applications and their input datasets (in order to allow for the re-computation of analyses) and on the level of data items (in order to allow for selective recomputations depending on an item-specific evaluation of impact predicates). Similar to approaches from the field of workflow provenance, the captured provenance information is represented through a graph that contains data nodes and edges to the consuming and producing operators, and sets of item-specific impact predicates. Overall, this information is less fine-granular than the information captured through the different forms of data provenance discussed above for relational queries. However, it suffices for the purpose of supporting the efficient recomputation of analysis processes, as discussed in this thesis.

6.3.3 Workflow Provenance

As stated in Section 2.5.2, the topic of provenance for (data-centric) workflows has received broad attention during the last decade. Many workflow management systems have included support for representing, persisting, and querying information about the provenance of the processed data. The surveys of [BF05, SPG05] provide an overview of the topic of provenance particularly in the field of scientific data processing, which has been a major application field of workflow management systems. As a consequence, most approaches to workflow provenance management address the derivation of data on the level of scientific data artifacts. That is, they capture the processing of, e.g., patient files, genetic sequence data, image data, and the like. The management of fine-grained provenance for large amounts of individual data records, such as the tuples of a database table, is usually out of scope for such systems. The *Taverna* system [MPB10] builds a notable exception in that it considers the provenance of large volumes of collection-based data, and provides fine-grained provenance capture and querying at the level of both collections and individual data items. The captured provenance trace of a Taverna workflow execution can be regarded as a directed acyclic graph, similar to the provenance graph described in Chapter 5. It comprises two types of relations, the first reflecting the data transformations that are computed by activities (processors) in the dataflow, and the second reflecting the transfer of data artifacts from the *output port* of one processor to the *input port* of the subsequent processor. The processed data artifacts of a Taverna workflow can be either atomic data items or collections. Thus, a processor may expect different types of data at a given input port, e.g., an atomic data item *x* or a collection of items *list(x)*. If the *depth* of the actual input data is larger than the expected depth at a given input port, this *depth mismatch* is handled implicitly. For example, if a processor expects an atomic input item ($depth = 0$), but receives a collection of items ($depth = 1$), a separate instance of the processor will be invoked over each element of the input collection. For capturing and querying the workflow provenance trace efficiently, Taverna uses this implicit iteration model. Thus, if required, provenance traces can be generated explicitly at the fine-grained item-level, or by using the implicit iterator model over collections. In the approach discussed in Chapter 5, a similar iteration model is used to implicitly iterate over datasets in the process of recomputing the results of analysis processes.

Open Provenance Model The *OPM* [MFF$^+$07] has emerged as a community-driven model for provenance capture as the result of the the Provenance Challenge series. It provides a technology-agnostic specification and a graphical notation of a provenance graph, which captures the sequence of transformations and the consumed and produced data artifacts of a given process. The OPM reflects basic concepts of provenance in a generic manner, and can be extended with application-specific concepts if required. It is largely and foremost applied in the field of workflow management; many systems such as [LAB$^+$06, MPB10, PCA11] provide implementations extending the model or im- and export capabilities to produce provenance traces from or to OPM.

In an OPM graph, different node types are used to represent *data artifacts*, *processes* using or generating such artifacts, and *agents* controlling the execution of those processes. Edges that connect those nodes within the graph can be of different types (*Used, WasGeneratedBy, WasTriggeredBy, WasDerivedFrom, WasControlledBy*) and annotated with role labels, depending on the nature of dependency between source and target nodes. For example, if an artifact *A1* has been generated by a process *P1*, this will be represented in the graph through an edge *wasGeneratedBy(A1,P1)*. Based on the graph model, the OPM further defines a set of inference rules and transitive closures over the dependency relationships. So, a user can investigate different questions such as which processes were used to generate a given artifact *A*, or from which base artifacts *A* has been derived.

While the provenance graph structure used in Chapter 5 of this thesis adopted the use of artifact and process nodes as well as selected edge types from OPM, OPM includes a large range of features that have not been considered in this thesis. For example, all OPM entities can further be annotated, e.g., with application-specific information. OPM also allow to capture temporal information about the existence of artifacts and the execution of processes, and defines temporal constraints that must be satisfied with respect to he (causal) dependencies of the provenance information. The use of so-called *accounts*, which enable the specification of multiple descriptions for the same process execution, and thus can be used to provide hierarchically refined views on the process provenance.

In general, the OPM addresses provenance on a much more coarse-granular manner than the previously discussed approaches to data provenance, due to its primary focus on the capture of workflow provenance rather than (relational) query provenance. On the other hand, the OPM captures the precise sequence and properties of applied transformation, and includes concepts such as roles, agents, or accounts, which are highly relevant in the context of workflow management, but are completely out of scope of the models for data provenance.

6.4 Systems

We now consider selected systems for uncertain data management. The goal of this section is to give an overview of how the various approaches discussed previously are applied by those systems, and how they differ from the concepts presented in this thesis. We broadly distinguish two classes of solutions. First, we discuss systems that focus *mostly* (though not exclusively) on the evaluation of relational queries over uncertain data in a data warehouse setting. Those solutions primarily consider large volumes of discretely distributed base data, such as occurs in the field of information extraction or data integration tasks. In this context, we discuss Trio [ABS+06], PrDB [SDG09] and MayBMS [HAKO09], all relying on an explicit representation of uncertainty. Then, we shift the scope to such approaches that focus both on the evaluation of relational operators as well as statistical analyses, and which are positioned in the area of what-if-analysis. This includes the Monte Carlo Data Base (MCDB) system [JXW+08] as a sample-first approach to uncertain data management, and the PIP [KK10] system, as well as the Jigsaw [KN11] approach which further extends PIP.

6.4.1 TRIO

The Trio system implements an integrated approach for uncertainty and lineage management, based on the model of Uncertainty and Lineage Database (ULDB), first discussed in [BSHW06].

Uncertainty and Lineage Database

The capture of source provenance, referred to as “lineage”, is the first central concept of Trio. To enable the handling of lineage, Trio associates each tuple in a database with a unique identifier (i.e., an annotation). A Lineage Database (LDB) is a triple $D = (\bar{R}, S, \lambda)$, where $\bar{R}$ denotes the set of relations, $S = I(\bar{R})$ denotes the set of identifiers for all tuples in relations $\bar{R}$, and λ is a lineage function $\lambda : S \rightarrow 2^S$ which associates with each tuple t the set of tuples that were used in its derivation. ULDBs extend LDBs by incorporating the concept of uncertainty. The ULDB model can be considered a specialization of the generic model described in [SBH$^+$09]. Trio uses the concept of *X-tuples* and *X-relations* to represent tuple alternatives and uncertain relations as sets of such alternatives. An *X-tuple* is a set of alternative tuples (representing disjoint events) while different *X-tuples* are considered *independent*. Each X-tuple alternative is assigned a confidence value in $[0, 1]$ which denotes its probability of being in a possible instance of the database; the sum of probabilities assigned to all alternatives of an X-tuple must be 1. Further, it is possible to define *Maybe X-tuples* as X-tuples marked with a '?', to indicate that its presence in the database is not certain. As a consequence, the sum of probabilities assigned to alternatives of a Maybe X-tuple can be below 1 (which relates to the representation of partial distributions in [SMS$^+$08]).

In a ULDB $D = (\bar{R}, S, \lambda)$ the symbols in $S = I(\bar{R})$ now correspond to tuple alternatives rather than tuples, as in LDBs. That is, any $s \in S$ is a pair (i, j), where i identifies the X-tuple and j refers to the j^{th} alternative of this X-tuple. From a ULDB, one can derive a possible LDB $D_k = (R_k, S_k, \lambda_k)$ by choosing exactly one (at most one) of its alternatives for every X-tuple (Maybe x-tuple) t_i. The associated symbol $s_{(i,j)}$ is included in S_k. Similarly, for every $s_{(i,j)} \in S_k$ its lineage $\lambda_k(s_{(i,j)})$ is included in S_k, since all tuples that contributed to the alternative associated with $s_{(i,j)}$ must be contained in the LDB as well. This approach again relates to the work of [SMS$^+$08], where the so-called *history* of tuples is handled likewise. Besides discrete distributions, Trio also supports for random variables distributed over continuous domains [AW09], which are represented symbolically.

Two-Phase Query Processing

To enable the incorporation of lineage information in the computation of confidence values for result tuples, the Trio system evaluates queries in two steps. Result confidences are computed in a *lazy* approach, i.e., only at the end of the evaluation when they are requested. In the first phase, the data computation is performed, i.e., the query result as such is computed, and lineage information is recorded. As discussed, the lineage information reflects dependencies between source and (intermediate) result data. This information is then used in the second phase of the query evaluation to calculate result confidences consistent with probability theory. More specifically, the lineage information of each result tuple t is used to trace back to the base data (tuples) from which t was derived. Then, the result confidence of t is computed from the probabilities of the base tuples.

Continuous Distribution Support

Trio supports the processing of continuous distributions, which are represented symbolically, using approximate processing techniques at query time. The system relies on the use of various approaches, such as accessing distributions through representative functions or using histogram-based approximations, in order to numerically integrate over or compute statistical measures from the continuous distributions (see Section 2.4.1). Examples for representative functions include the function $mass(min, max)$ that returns the integral of the distribution in $[min, max]$, and the function $sample(min, max)$ that generates a random sample within a given support interval $[min, max]$.

In this thesis, continuous distributions are represented in a similar fashion to Trio by relying on symbolic or histogram-based representations and providing basic operators to access the represented distribution information. However, opposed to the central approaches discussed in this thesis, Trio focuses on general relational query processing over volumes of (mainly) discretely distributed data. Trio does not address techniques for modeling joint distributions between variables, which is addressed in this thesis through the ACR approach. Also, Trio itself does not incorporate methods for an efficient re-evaluation of queries under changed inputs, as addressed in the context of analysis processes in this thesis. Later work of the authors of Trio, such as [IW10, ISW10], addresses the topic of (selective) query re-evaluation for relational queries based on recorded predicates, yet in the context of deterministic rather than probabilistic data.

6.4.2 PrDB

The PrDB system, described in [SDG09], provides a flexible probabilistic data model that supports the specification of uncertainty at tuple and attribute level and the representation and querying of complex correlation structures. While [SD07] focused on the representation of tuple-level uncertainty, [SDG09] includes the handling of attribute-level uncertainty.

Each relation R in the PrDB database is associated with (probabilistic) attributes $attr(R)$ and consists of a set of (probabilistic) tuples t which map each probabilistic attribute value in $attr(R)$ to a random variable. The variable corresponding to $t \in R$ and $a \in attr(R)$ is denoted as $t.a$. Every tuple t of R is further associated with a boolean random variable $t.e$, which represents the tuple's existence in the database D. Using $\mathcal{X}$ as the set of all random variables associated with relations in D, one can produce a deterministic database (i.e., a possible world) through a complete assignment of values from $dom(\mathcal{X})$. Based on this model, the special focus of PrDB is the handling of correlations between random variables, which can be both introduced during query evaluation and present in source data. Since PrDB primarily targets domains such as information extraction and (sensor) data integration, its focus is on the efficient representation and evaluation of joint distributions over large sets of discrete random variables. Contrary to the focus of this thesis, the handling of arbitrary correlation structures between continuous marginal distributions is not in the scope of PrDB.

Probabilistic Graphical Models in PrDB

PrDB uses a PGM representation in order to represent arbitrary dependency structures between (potentially large numbers of) discrete random variables. Thus, a probabilistic database $D = (R, P)$ is defined by a set of relations, R and a corresponding PGM P. Joint distributions are represented using a factored representation. Recall that factors represent interactions within sets of (dependent) variables in a graph while independent sets are associated with different factors. Therefore, they allow for a compact representation of complex joint distributions in PrDB. For example, factors can express dependencies such as mutual exclusion, implication, or independence. Instead of storing the information about the uncertainty and dependencies directly with the data, PrDB applies the concept of separate *partitions*. Each partition stores a factor together with references to those variables that are arguments of the factor; conversely, with each individual variable the system stores references to the partitions from which the variable is referenced.

The use of factors and the concept of a partition can be viewed analogously to the relation of a joint distribution x_{jnt} and its components, as discussed for the representation of multidimensional distributions in Section 3.2.2. In particular, factors f in the PGM relate to correlation components $x.cor$, while the individual variables that are arguments to f relate to the dimension components $x.dim_j$.

Query Evaluation and Optimization

PrDB supports a wide range of queries including SQL queries, inference, and decision support queries. As described above, one benefit of the PGM approach is the possibility of applying efficient query evaluation techniques, even when complex correlation structures (with a large number of variables) are involved. For example, PrDB employs the VE inference algorithm to compute a marginal distribution of a variable X_i from a joint distribution. Still, inference can get highly complex if the PGM contains factors that involve large numbers of variables, e.g., as a result of an aggregation. For example, when n tuples are aggregated into a result tuple t, this produces a large factor involving $n + 1$ arguments. As one approach to mitigate this performance problem, the PrDB system exploits the *decomposability* of many operations, such as the union or aggregation operations. Potentially very large factors are thus split up into a linear number of constant-sized factors, thereby allowing an efficient evaluation over the complete joint distribution. For further optimization, PrDB resorts to *approximate* evaluation approaches, such as MCMC techniques, whenever exact query evaluation gets overly expensive.

Correlation Modeling through Factors and ACRs

While PrDB typically considers a low number of discrete possible values per variable, this thesis focuses on the case of continuously distributed variables. As a result, joint distributions between those variables theoretically reflect an infinite number of conditional probability values. In practice, we apply m-dimensional histograms (ACRs) to approximate the underlying correlation structures, which are represented by copula functions. This use of ACRs (copulas) implies a different approach to dependency modeling and processing than the PGM approach, and the involved number of correlated variables is typically assumed to be much lower. As stated before, one could still consider an ACR analogous to a factor in a PGM. A factor representing the same joint distribution as an ACR with a representation granularity of β and m dimensions would contain β^m conditional probability values, assuming there is *no conditional independence* between the m correlated variables. That is, the overall size of the factor depends on the granularity of the distribution approximation, β, and the dimensionality m, similar to the ACR-based representation. As stated in 6.2, PrDB can generally represent and process arbitrary joint distributions similar to the ones that can be *produced* using ACRs. However, the ACR approach differs from the PGM approach in that it provides the benefit of modularity: While PrDB models every joint distribution between the *concrete* variables, the self-contained correlation representation of ACRs is decoupled from the actual data. Thus, it can be used to correlate several variables with arbitrary univariate distributions, and compare the resulting joint distributions. This is more appropriate for application scenarios in which a user wants to evaluate complex correlation structures and compare their influence on future business developments, e.g., for the purpose of risk analyses.

6.4.3 MayBMS

The MayBMS system was developed as a complete probabilistic database system extending the Postgres server backend. MayBMS uses *U-relations*, a version of probabilistic c-tables (see Section 6.2), to represent probabilistic data with dependency information. The U-relations of MayBMS provide the same expressiveness as the PGM approach used in the PrDB framework. Concerning the representation and processing of joint distributions this means that MayBMS similarly enables the user to represent any joint distribution between arbitrary discretely distributed variables. Again, this representation approach is different from the ACR approach taken in this thesis, which distinguishes itself through the self-contained representation of correlation structures.

The MayBMS system features a query and update language extending SQL for processing and transforming probabilistic data. Based on the U-relational representation, MayBMS provides for efficient query processing through probabilistic inference techniques.

Query Algebra

The query algebra of MayBMS is the *probabilistic world-set algebra* (probabilistic WSA). It contains the relational algebra operations (σ, Π, $\times$, $\cup$, $\setminus$, and ρ), an operation *conf(R)* for computing the confidence of result tuples of a relation *R* and a *repair-key* operation for introducing uncertainty.

The operation *conf(R)* computes the confidence for each tuple in relation R as the sum of probabilities for all possible worlds in which the tuple occurs. The resulting (certain) relation contains an attribute P which keeps the total confidence value for each tuple. Further, operations *poss(R)* and *certain(R)* are provided; they are computed as a selection on the results of *conf(R)*, to yield those tuples that are possible (i.e., $P > 0$) or occur with certainty (i.e., $P = 1$), respectively.

The operator *repair-key*$_{\vec{A}@B}(R)$ conceptually introduces uncertainty to the database. It does so by deriving a maximum repair of relation R with respect to a functional dependency $\vec{A} \rightarrow schema(R)$. That is, the result of applying *repair-key* is the set of possible worlds S_j, each of which is a relation that satisfies the functional dependency $\vec{A} \rightarrow schema(R)$. Each such world is assigned a probability p_j computed from the weight values associated with the contained tuples through a numerical attribute B. Apart from introducing uncertainty to an initially certain database, the *repair-key* operation further enables the computation of a database of *posterior probabilities* from a database with prior probabilities based on new evidence data.

In this thesis, in contrast to the *repair-key* operator, uncertainty is only introduced to data by deriving distributions of a (newly introduced) random variable from historic fact data. This can be done either by computing parameters of a distribution function, or by constructing a histogram representation from underlying fact data. The *repair-key* operation has a different focus and is more generic in that it allows the introduction of uncertainty (and correlation information) based on both initially certain and uncertain data. Different to MayBMS, this thesis interprets uncertainty in terms of random variables over continuous domains. Thus, the focus was put on the derivation of continuous distributions, as well as the evaluation of correlation structures (ACRs) between continuously distributed variables.

Query and Update Language

The query and update language of MayBMS is a generalization of SQL over traditional relational databases. For so-called *typed-certain* U-relations (classical relational tables), MayBMS supports full SQL. For queries over *uncertain* U-relations (i.e., U-relations that contain random variables and thus are uncertain) MayBMS applies some restrictions, in particular with respect to aggregation. MayBMS replaces the standard SQL aggregates (*sum* and *count*) by operations for computing the *expected* aggregate (*esum* and *ecount*), which allow for an efficient processing and representation of aggregates over U-relations. This relates to the semantics applied for the aggregation operator $\mathcal{A}_{SUM}$ in Section 3.3.3. The computation of confidence values (*conf(R)*) is also implemented as an aggregate in the MayBMS query language. Similar to the aggregation functions presented in this work, the aggregation operators of MayBMS compute aggregate values that are *deterministic*, i.e., each of the aggregate operations in MayBMS produces typed-certain output tables.

MayBMS further supports SQL *data manipulation* queries for inserting, creating, and deleting data both for typed-certain and uncertain U-relations. As a special update operation, MayBMS provides the *assert* operation. This operation serves to *condition* a database using a constraint Φ (expressed through a boolean positive relational query), such that the result database contains only those possible worlds that satisfy Φ. This assert operation can be applied, for example, to introduce integrity constraints over an uncertain database and thus reduce the contained uncertainty, e.g., for the purpose of data cleansing. The operation of conditioning relates to the computation of conditional marginal distributions over continuously distributed variables. There, the density in the region that satisfies the applied conditions is similarly normalized to 1, while the region that does not satisfy the applied conditions is discarded from the further analysis.

The preceding sections have discussed three systems that rely on an explicit representation of discrete probabilistic data. Next, we consider two solutions that rely on MC simulation and support the representation of arbitrarily distributed data.

6.4.4 MCDB and MCDB-R

As its name suggests, the Monte Carlo Data Base (MCDB) [JXW+08, XBE+09] applies MC sampling on the database level to *implicitly* represent and process probabilistic data. MCDB allows users to specify queries involving attributes that are generated through so-called Variable Generation (VG) functions. Those functions, and the parameter tables used to parameterize them, encapsulate all information about the uncertainty in data.

Random Relations and VG Functions

In MCDB, a random relation (i.e., a relation that contains at least one uncertain attribute) is specified by a schema and a set of VG functions which are used to (pseudo-randomly) generate the samples for a number of MC iterations. An extended version of the SQL `CREATE TABLE` syntax is used to identify VG functions and the corresponding parameter tables which are to be used to parameterize the desired VG function in the query. Users can choose from a set of default VG functions (e.g., Normal, Gamma), or use custom-implemented VG functions.

MCDB follows the assumption of [DS07] that each random relation R can be viewed as a union of blocks of correlated tuples, where tuples in different blocks are *independent*. In each iteration of the simulation, a set of samples is generated for all random attributes in the query, corresponding to a possible world in the database. Ordinary relational queries, such as selection, join, grouping, and aggregation, are then evaluated over the samples. The query result is an empirical distribution which can then be evaluated further by the user. He can visually analyze this distribution, or further evaluate it by computing statistical measures over the distribution.

As an example (adopted from [JXW+08]) suppose a user wants to prospect sales for the coming year based on a forecast of per-customer sales. He does not know the exact sales amount that will be generated per customer, but he has available historic data about the customers region. He assumes the sales to be Gamma distributed, with the *scale* parameter depending on the region the customer lives in, and the *shape* parameter being determined for each customer individually. The parameter information is made available to the VG functions through input parameter tables. Listing 6.1 shows the query implementing this example.

Listing 6.1: MCDB Example Query

```
CREATE TABLE SALES(CID, AMOUNT) AS
  FOR EACH c in CUSTOMERS
   WITH AMOUNT AS Gamma(
    (SELECT sh.SHAPE FROM AMOUNT_SHAPE sh
     WHERE sh.CID = c.CID),
    (SELECT sc.SCALE FROM AMOUNT_SCALE sc
     WHERE sc.REGION = c.REGION))
   SELECT c.CID, m.VALUE FROM AMOUNT m
```

In the query, two parameter tables `AMOUNT_SCALE(REGION, SCALE)` and `AMOUNT_SHAPE(CID, SHAPE)` are provided to the VG function `Gamma`. The `FOR EACH` clause instructs MCDB to loop over the list of customers stored in the `CUSTOMERS` table and to generate a realization of the `AMOUNT` attribute for each customer via a call to the `Gamma` function. This function returns a one-column single-row table (`AMOUNT`) as result. The sample data thus created (for n MC

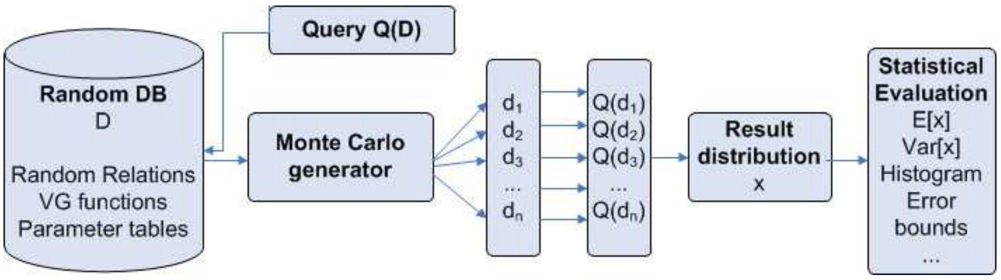


Figure 6.1: Query evaluation in MCDB (adopted from [JXW+08])

iterations) represent n possible worlds, which are then evaluated in a deterministic fashion. Thus, the query `SELECT SUM(AMOUNT) AS totalSales FROM SALES` yields the prospected total sales amount in the form of an *empirical distribution* computed over the sampled possible worlds.

Query Processing

Figure 6.1 illustrates how query processing proceeds in MCDB. First, the VG functions specified in a query Q, such as the `Gamma` function used above, are invoked with the provided parameter tables, generating a set of n i.i.d. samples for the corresponding attributes. Those samples correspond to n possible worlds d_j. The *relational* part of the query is then evaluated over each of those possible worlds, resulting in a set of n results $Q(d_j)$ contributing to the query result distribution x. This resulting empirical distribution can then be input to further *statistical evaluation* functions, to derive statistical characteristics, such as its variance or expected value.

Since a random relation in MCDB conceptually consists of sets of many hundreds or thousands of possible worlds (depending on the number n of MC iterations), the execution and processing times of a query are crucial. In a naive approach, each possible world would be materialized for its own, resulting in very high CPU and memory requirements to store all the emerging tuples. To mitigate this problem MCDB makes use of the fact that all *deterministic* values are the same in each possible world. MCDB thus uses *tuple bundles* to represent random tuples succinctly. In a tuple bundle, each deterministic attribute is represented only once and each random attribute is included as a nested vector that contains the n different sample values. The following example shows a tuple bundle `t` for the previous example:

```
t = (CID, AMOUNT, PRESENT) = (42, (123.50, 274.00, ..., 180.76), isPresent)
```

In the tuple bundle, the values of the random `AMOUNT` attribute are stored as a nested vector, while the deterministic `CID` attribute is stored only once. Queries are executed over tuple bundles instead individual tuples, which means that each deterministic attribute is evaluated only once, and only the nested vectors incur multiple evaluations (one evaluation per sample value). Optionally, if not all costumers in our `SALES` table always appear in each possible world (e.g., due to an applied predicate), a boolean vector `PRESENT` is used to further optimize query processing.

Recomputability

Since pseudo-random number generation is a deterministic process when started from a fixed seed, one can generate an equal set of samples based on the same seed. MCDB enables the recomputation of queries by capturing and reusing the seed values applied by the VG functions in the initial computation. Hence, the tuple bundle from the previous example can be stored in compressed form as `t = (42, seed, isPresent)`, where `seed` is a single seed value. The values for the random attributes, in this case `AMOUNT`, can then be regenerated in a consistent manner when required—given the applied VG function and parameterization is accessible.

MCDB-R: Support for Risk Analysis

In the context of risk analysis, [JXW$^+$10] describes an extension to the initial MCDB approach. MCDB-R targets to support functionalities such as the sampling from *extreme* (i.e., very low or high) quantiles of query result distributions. MCDB-R primarily follows the goal to increase the *sampling efficiency* for such queries, especially in the case of very light-tailed result distributions. The problem that occurs when sampling such distributions is that many samples will eventually be discarded since they do not meet the selection criteria. For example, for a very light-tailed distribution only very few samples may exceed a specified upper tail threshold; thus, a large number of samples from the remaining domain of the distribution will be discarded, and a lot of effort is wasted during the sampling phase.

To enable a more efficient approach to sampling from the tails of a result distribution, the MCDB-R system applies the so-called *Gibbs cloning* approach. This approach combines Gibbs sampling [GG84] and *cloning*, a technique from the area of *rare event simulation*. The idea of cloning is to take a low number of initial randomly generated rare events (called *particles*), and produce further extreme samples as clones of those particles. The cloned samples are created based on a Markov chain that is constructed through stepwise random perturbations from the initial particles; only those clones that are indeed extreme are then retained and added to the result sample set.

In the MCDB-R system, the Gibbs cloning approach is implemented efficiently through the so-called *Gibbs looper*. The tuple-bundle approach from [JXW$^+$08] is also extended to account for additional bookkeeping information that is necessary to support the Gibbs cloning approach. Using those extensions, the MCDB-R achieves considerable speed-ups for processing risk analysis queries, particularly in the case of light-tailed result distributions.

MCDB in the Context of Scenario Analysis

Overall, the application focus of MCDB and MCDB-R are in the field of what-if analysis and risk evaluation, and their requirements are closely related to our work.

In both systems, the specification of uncertainty and correlation structures are completely encapsulated in the VG functions. In addition to the provided default VG functions, users may implement their own sampling functions. Thus, one can basically model any desired joint distribution, provided appropriate statistical functions are available or have been implemented. On the one hand, the great flexibility is the principle benefit of the MC approach. On the other hand, encapsulating the complete information about the stochastical models in VG functions may lead to a loss of maintainability and reusability.

We hold that the application of copulas (ACRs) offers a more modular approach to constructing multidimensional distributions flexibly than does the implementation of *specific* joint distributions through VG functions. In Section 7.7.1, we discuss how the copula approach can be integrated with MC approaches such as MCDB. There, we also address the downsides of the native sample-based implementation as opposed to the representation through ACRs. Similar to the optimization approaches of MCDB-R, this thesis addresses the efficient evaluation of risk-related measures, such as joint tail probabilities of a distribution. In this respect, the use of ACRs as precomputed correlation structures enables a high query result accuracy, since the ACR structures can be computed offline at any desired degree of accuracy, using a potentially very high number of samples, if required.

While MCDB also focuses on the computation of what-if analyses, it does not address the capture and evaluation of provenance information over the created data. Thus, apart from enabling the recomputation of queries based on the retained seed values, the reuse of previous computations is not accounted for by MCDB. In contrast, this thesis addressed the capture of provenance as well as an approach for the re-computation of analysis results based on deviations in the input data.

To conclude, the very generic approach of MCDB is very well suited for computing stochastic models at any desired complexity, yet at the drawback of a potentially lower maintainability and high runtimes due to the costs of sampling. As will be shown in Section 7.7.1, the copula approach for handling arbitrary correlation structures can be integrated with MCDB in order to improve the analysis of low-dimensional joint distributions with arbitrary correlations.

6.4.5 PIP and Jigsaw

The PIP system extends the capabilities of MayBMS by supporting uncertainty in the form of both discrete and continuous distributions. Continuous distributions are represented symbolically and instantiated at runtime using MC simulation. In contrast to MCDB, PIP specifically aims to accelerate query processing by decreasing the effort spent for sampling, or trying to sidestep the sampling phase completely. In brief, PIP achieves this acceleration by deferring sampling to the final phase of query evaluation, and by exploiting specific information about the distributions to be sampled.

Probabilistic C-Tables with Symbolic Representations

For the representation of uncertainty, PIP uses probabilistic c-tables similar to MayBMS, allowing random variables in the c-tables to be associated with symbolic representations of continuous distributions, similar to [JXW$^+$08]. The symbolic representations are likewise specified in the form of distribution classes (e.g., Normal or Uniform), which are associated with random variables. For example, $X \rightarrow Normal(10, 1)$ states that the variable X is associated with a Normal distribution with parameters $\mu = 10$ and $\sigma = 1$. Additional information, such as the definition of the PDF or CDF, can optionally be provided to the system. A function $p(X = x, Y = y)$ specifies the joint distribution of X and Y (through a probability mass or density function for discrete or continuous variables, respectively). Similar to the VG functions of [JXW$^+$08], PIP supports a set of standard distribution functions and alternatively allows users to implement arbitrary required distribution functions. Functions of random variables are themselves considered as (composite) random variables.

Query Evaluation

PIP performs the sampling of random variables involved in a given query in a goal-oriented fashion. Sampling is delayed in order to reduce the effort wasted for samples which would not be considered during the subsequent evaluation. That is, PIP first evaluates the relational part of a query—as far as possible—over the c-table representation. For example, selections, joins, and grouping based on non-probabilistic columns are performed first. The result of the first step is again a c-table representation. If inconsistent conditions exist in those c-tables (such conditions can be introduced during query processing), the associated tuples are removed (as their conditions can never evaluate to *true*).

For the second phase of query evaluation, PIP applies various sampling techniques. The symbolically represented distributions associated with the *remaining* random variables of a query are thus instantiated. Finally, histograms or statistical moments are computed over the resulting samples. To increase the efficiency of the second evaluation phase, the system exploits additional information that can be provided for each distribution function, such as its (inverse) CDF. Depending on the availability of such information, the system can choose an optimal technique for computing statistical measures. This can help to speed up or even skip the sampling step, thus increasing the overall efficiency of query evaluation. For example, given information about the CDF associated with a variable X, the system requires only two function calls to determine the probability that a sample for X will fall within a specified interval. Further, PIP applies a number of optimized sampling techniques in order to decrease the overhead of basic techniques such as rejection sampling. For example, if a predicate ω is applied to values of a random variable X, PIP can use the inverse-transform approach (see Section 2.4.1) to sample directly from the part of the domain of X that satisfies ω, given the ICDF of X is known. This relates closely to the use of the inversion approach for efficient processing of correlation information in Chapter 4. PIP further exploits information about the independence between variables by subdividing constraint predicates into minimal subsets (i.e., such sets that do not share any variables). Then, it starts the sampling process for each of those subsets independently, thereby reducing the total number of samples required in total.

Jigsaw and Fuzzy Prophet

The Jigsaw system presented in [KN11] extends the approach of PIP to support the efficient computation of what-if scenarios over uncertain data. The specific focus of Jigsaw is to solve optimization problems, i.e., to find the optimal parameterization for a stochastical model with respect to a target function. The simulation process of Jigsaw proceeds by evaluating a specified what-if scenario—defined through a query over a parameterized stochastic model—for a number of parameterizations, and selecting the parameter setting that yields the optimal query outcome. In a basic approach, the query would need to be evaluated for every parameter setting in a potentially large parameter space. However, this yields prohibitively large processing times for even relatively simple models. Jigsaw addresses this performance problem by identifying correlations (similarities) between the parameterization and the output of a

VG function, and *reusing* the result of VG functions where possible. To this end, the system applies the so-called *fingerprinting* technique.

The basic intuition of the fingerprinting concept is that the outcomes of many stochastic functions are strongly correlated under *different input parameter settings*. Jigsaw enables the automatic identification of such correlations between function outputs and exploits the identified correlation in order to reuse the output of a preceding VG function invocation as the result for a correlated point in the parameter space.

Jigsaw uses *fingerprints* θ as a basis to determine similarity between two functions or between the outputs of one function F under different parameterization P_i and P_j. Fingerprints concisely represent the output distribution of $F(P_i)$ as a small ordered set of outputs of $F(P_i)$. The use of fingerprints for comparing the similarity of two functions relies on the approach of *random testing* [Ham94]. In random testing, deterministic functions are compared based on the pairwise similarity of their results given random input values. Jigsaw transforms *stochastic functions* F to deterministic functions by replacing all sources of randomness in *both* invocations of F using the same vector σ of m seed values. Then, the resulting fingerprints of two stochastic functions $F(P_i), F(P_j)$ are defined as the m outputs $F(P_i, \sigma_k), F(P_j, \sigma_k)$ that are produced based on the common seed vector $\sigma = \{\sigma_k\}$.

To determine similar fingerprints, Jigsaw aims to find a closed-form mapping function $\mathcal{M}$ that maps individual elements of a fingerprint θ_i to the corresponding element in θ_j. During the query execution process Jigsaw maintains a set of *basis distributions* that associate fingerprints θ_i with a previous output o_i. Whenever the fingerprint θ_j of a new function parameterization P_j is similar to a previous fingerprint θ_i, the output for P_j can be computed as a mapping of o_i. Else, $F(P_j)$ is computed and inserted as a new basis distribution. This approach enables Jigsaw to reuse results whenever a correlation between the parameterization and outputs of a function can be identified. Thus, Jigsaw increases the efficiency of the PIP approach further, decreasing query runtimes considerably when large numbers of parameterizations need to be evaluated.

Jigsaw further provides the tool *Fuzzy Prophet* that enables an *interactive exploration* of the parameter space. A user may specify a region of interest in the parameter space through a graphical user interface; Jigsaw then rapidly produces accurate results for the selected parameter settings based on a very small fingerprint, and generates further samples to *refine and validate* the accuracy of the result and the maintained basis distributions. To speed up potential future explorations, Jigsaw heuristically generates fingerprints for adjacent regions in the parameter space that may be relevant to the user, thus increasing the efficiency of the further exploration steps.

While Jigsaw aims at an efficient computation of scenarios based on similar input parameterizations, the recomputation approach discussed in this thesis aims to exploit the *explicit* representation of deviations in input data. Although both approaches exploit provenance information, the sort of information and the way this information is used are vastly different. This is due to the different underlying representation paradigms, i.e., the implicit uncertainty modeling of Jigsaw (and the base system PIP) and the explicit approach applied in this thesis. Therefore, the two optimization approaches are not directly comparable, but should rather be considered as alternative approaches that are tailored towards each of the specific representation forms. A graphical user interface that supports the interactive exploration of the scenario space, such as provided by Fuzzy Prophet, has not been implemented during the course of this thesis.

After the different solutions to uncertain data management have now been discussed, the next section will once again summarize their main distinction to the approaches discussed in this thesis.

6.5 Conclusion

This section concludes the discussion of related work by summarizing the relation of the presented systems with the contributions of this thesis. For a comprehensible overview, Table 6.4 summarizes the main aspects of the discussed solutions regarding the *modeling* and *processing* of data. Regarding the data modeling aspect, we consider the distinction of *explicit* and *implicit* representations of probabilistic data, the type of distributions that are supported (i.e., no support (-), weak support (+), and strong support (++) for representing *discrete* and *continuous* distributions), and the modeling of *dependency (correlation) and provenance*. The latter comprises aspects of *query processing*, supported *analysis functions*, and support for the *recomputation* of analysis queries. The table also lists the implementation of the concepts presented in this thesis (see the gray-shaded row), which will be discussed and evaluated during the subsequent chapter. This implementation is denoted as *HANA Ext*. In line with the previous rough categorization of approaches, *HANA Ext* is compared against the approaches based on the *explicit (model-extension)* and *implicit (MC-based)* representation of uncertainty, respectively.

Model-Extension Approaches

Similar to the Trio, PrDB, and MayBMS systems, the approaches discussed in this thesis rely on an explicit representation of uncertainty. However, in contrast to the named systems, the focus is on the representation of *continuous* distributions represented in histogram-based or symbolic form. While Trio provides some support for continuous distributions, their support is restricted to a selected number of distribution families, and thus to the representation of data that follows a suitable distribution. In contrast, this thesis also allows for the representation of arbitrary distributions through histograms. Parameterized distribution functions are accessed through a provided definition of their PDF or CDF, or processed based on a discrete (histogram-based) approximation at runtime.

Another distinction can be made with respect to the handling of dependencies in data. While Trio handles dependencies introduced in the course of query processing, base data is generally assumed independent and the Trio system does not allow to specify any dependency information. The expressivity of PGMs and probabilistic c-tables, as used in the PrDB and MayBMS systems, generally allows for representing arbitrary dependencies both in base and (intermediate) result data. This enables the representation of arbitrary joint distributions. However, in contrast to the concepts for correlation handling presented in Chapter 4, PrDB particularly focuses on the handling of correlation structures over a large number of variables over *discrete* domains. Similarly, MayBMS expresses joint distributions between discrete variables based

System	Modeling			Processing		
	Representation	Discrete/ Cont.	Dependency & Provenance	Query Processing	Analysis Functions	Recomputation
Trio	Explicit (X-relations)	++/+	Independent base data, 'Trio-lineage' on tuple-level:	Two-phase: Lazy confidence computation, cont. distributions through characteristic functions, sampling	Computation of moments, threshold & top-k queries	—
PrDB	Explicit (PGM)	++/–	Dependency encoded in PGM, factored representation, partitions	Exact (VE) & approximate (MCMC) inference over PGMs, graph decomposition	Computation of conditional joint probabilities, marginal distributions	—
MayBMS	Explicit (c-tables)	++/–	Conditions over variables	Query plans over c-tables	Repair-key, computation of confidence, posterior probability, conditioning	—
HANA Ext	Explicit (Variables w/ histograms)	+/++	Decoupled correlation information (ACRs), provenance at transformation- and data item-level	Query plans w/ custom operators, histogram approximation, approximate inversion-based correlation processing	Correlation introduction & extraction, computation of conditional joint probabilities, marginal distributions, (temporal) aggregates	Selective recomputation based on impact relevance
PIP	Hybrid (Prob. c-tables, variables w/ VG functions)	++/++	Conditions over variables, multivariate VG functions	Two-phase: Query plans over c-tables, goal-oriented sampling, exploiting independence	Statistical aggregates, conditional probabilities	—
Jigsaw				Capture & evaluation of fingerprints during processing		Fingerprint-based reuse of results
MCDB	Sample-First (Variables w/ VG functions)	+/++	Multivariate VG functions	Two-phase: Sampling & deterministic query evaluation over sampled PWs	Aggregation, moments	
MCDB-R				Gibbs sampling	Efficient computation of extreme tails, quantiles	—

Table 6.4: Overview of characteristic features of related approaches

on the U-relational representation. The handling of *arbitrary* correlation structures involving *continuous distributions* is neither in the scope of PrDB nor MayBMS.

In general, none of the discussed approaches account for the extraction, storage, and introduction of arbitrary correlation structures between arbitrary distributions. The approaches addressed in Chapter 4 address exactly this open issue by means of ACRs, which can be used to represent correlation structures that are either extracted from historic data or precomputed, and to introduce such correlation structures to arbitrarily distributed variables.

A related topic is the representation and use of provenance information by the different approaches. In the Trio and PrDB system, the *source provenance* of data is captured as the *lineage* of tuples or in the form of *factors*, respectively. There, the primary goal is to enable the correct computation of confidence values associated with relational query results. The MayBMS system does not yet include support for provenance handling, but U-relations generally can be used to reflect such information. In contrast to the capture of fine-grained (tuple- or attribute-level) source provenance for relational operators, Chapter 5 of this thesis discusses the capture and use of both source and transformation provenance for analysis operators. The provenance information is captured at different levels of granularity, i.e., on the level of data sets and items as well as operator applications. Both sorts of provenance information are captured with the goal to enable the efficient recomputation of analysis processes under deviations in the input data.

The overall aim of Trio, PrDB, and MayBMS is to provide a probabilistic database for evaluating generic *relational queries* over large amounts of uncertain data. Their application context, as stated in most of the publications related to the systems, is largely situated in the fields of sensor data management, information extraction, and data cleaning and integration. MayBMS explicitly also addresses the field of (what-if) analysis, providing functionality for introducing uncertainty (*repair-key*) and computing posterior probabilities based on new evidence data, as well as conditioning a database. In contrast to the above-named solutions, this thesis has addressed primarily the execution of analysis processes involving analytical operators, rather than providing a complete probabilistic data management system. For example, support for joining and grouping data based on uncertain attributes is not provided for by this thesis. The application of generic relational queries over fact (and thus, deterministic) data is assumed as a step that precedes the further analysis. Thus, the analysis operators supported in this thesis are in most cases applied on previously joined, grouped, and aggregated input fact data. The aspects of correlation handling and recomputations within the analysis process have been of primary interest in this thesis. Both aspects have not been addressed appropriately in prior work, yet are highly relevant to support flexible scenario analysis and planning processes.

Monte Carlo Approaches

The *MCDB and PIP* systems (and their extensions *MCDB-R* and *Jigsaw*) constitute solutions that rely on the implicit approach to uncertain data management, using MC simulation. Similar to this thesis, both the MCDB and PIP approaches mainly address the application context of statistical analysis and what-if queries over large volumes of data. They include functionality for the introduction of uncertainty (hypotheses) in data and the evaluation of statistical measures, such as the mean or variance of query results. While MCDB implements the pure sample-first approach, PIP employs a more goal-oriented and thus more efficient sampling, by deferring the sampling phase until after the relational part of a query has been evaluated as far as possible. Using MC simulation, both systems provide for constructing and querying complex statistical models of theoretically arbitrary complexity. In this respect, one can argue that MC-based solutions are the most generic solutions to uncertain data modeling, since any desired distribution can be represented given suitable (custom) VG functions are provided. However, as previously stated, the potential complexity of the resulting stochastic models may reduce maintainability and can lead to high runtime requirements in case optimized sampling techniques are not supported or not applicable. Although both MCDB and PIP enable the modeling of arbitrary joint distributions using custom-implemented VG-functions, they do not consider correlation as a self-contained artifact and do not address the extraction and introduction of correlation between arbitrary marginal distributions. In Section 7.7, a technical integration of the correlation introduction operator with systems such as MCDB and PIP will be described. At the same time, the drawbacks of the native (sample-based) implementation by means of a basic VG function will be discussed.

In addition to the concepts of MCDB, PIP particularly addresses the goal-oriented execution and optimization of the sampling phase during query evaluation. To this end, PIP exploits the availability of additional information such as the CDF or ICDF of a distribution. This optimization relates to the exploitation of the inversion technique in the implementation of $\mathcal{C}^{reverse}$, as well as the rewriting-based optimization of multivariate analysis queries, as discussed in Sections 4.4 and 4.5.

MCDB and PIP provide no functionality for efficient recomputations based on modifications in input data. Thus, changes to parts of the input data require a complete re-evaluation of the query in contrast to the selective re-evaluation discussed in this thesis. The Jigsaw approach addresses the problem of scenario computations under many different input parameterizations of the underlying stochastical models. Jigsaw seeks to achieve an efficient exploration of scenarios by exploiting knowledge about similarities in the output of VG functions, and enables an interactive parameter space exploration through the Fuzzy Prophet tool. Both the efficient scenario computation as well as the parameter space exploration enable a highly flexible approach to scenario analysis. This thesis, similarly, includes an approach to increase the efficiency of analysis process recomputations by selectively re-evaluating analytical operators depending on deviations in the input data. The applied approaches differ from those of Jigsaw both regarding the underlying representations of uncertainty (model-extension vs. MC-based) and the different focus of the applied techniques. While Jigsaw considers recomputations under different function parameterizations, the recomputation approach of Chapter 5 selectively recomputes operators for explicitly represented deviations in input data.

In summary, the application context of this thesis is close to that of the MCDB and PIP systems, due to their similar focus on scenario (what-if) analysis and planning. On the other hand, the explicit representation of uncertain data, their processing, and the capture of provenance are more closely related to the model-extension approaches discussed above. Different from all discussed systems, this thesis investigated the use of a self-contained representation of correlation structures, including their efficient extraction from and introduction to data. The selective recomputation of analytic queries over continuously distributed data constitutes an extension to previous work in the field of deterministic and discrete probabilistic data. The motivation for enabling efficient recomputations is close to the fingerprinting-based approach of the Jigsaw system, and can benefit the efficient evaluation of scenarios in the scenario planning process.

After having given an overview over related approaches and systems in the field of uncertain data management, the next chapter evaluates the concepts discussed in this thesis based on their prototypical implementation.

7 Evaluation

This chapter presents the experimental evaluation of the analytic operators discussed in Sections 3.3, 4.3, and 4.4, and the performance effect achieved by means of the provenance-based recomputation technique discussed in Chapter 5. The first section describes the prototype system that implements the discussed functionality. This system extends the SAP HANA database; it is therefore denoted as *HANA Ext* during the remainder of the chapter. Additionally, Section 7.7 discusses an extension to the MC-based systems MCDB and PIP by means of a custom VG function for correlation introduction. Finally, an evaluation of operators and queries performed on *HANA Ext*, MCDB and PIP is provided to compare the implementation of *HANA Ext* to the MC-based solutions with regard to their applicability and expressivity as well as the associated runtime requirements.

7.1 Implementation

This section describes the prototypical implementation of the *HANA Ext* system as an extension to the *Calculation Engine* (CE) component of the SAP HANA in-memory database.

7.1.1 Calculation Scenarios

The CE component of SAP HANA provides support to construct, parameterize, and execute so-called *Calculation Scenarios* (CS). A CS models complex calculations over data residing in tables of the database by means of a directed acyclic graph of *Calculation Views* (CV). A CS can be created, executed, reused, and extended by users. In the context of this thesis, a CS represents the implementation of an analysis process comprising several analysis steps. Intermediate or final processing results within such an analysis process are represented by means of a CV. A CV is being specified through a name, a set of input and output attributes, and an operation that transfers inputs to outputs. Input attributes can either be the output attributes of another CV or the attributes of a base table (i.e., a collection of columns physically stored in the database). To refer to a base table as input data one specifies a *Data Source (DS)*. The DS specification contains information about the physical input table as well as the exposed columns and a name by which the DS is referenced.

Similar to the way the SELECT-clause of an SQL CREATE VIEW statement refers to other tables or views, one can specify a CV by referring to one or several input CVs and DSs, and by specifying the operation that is applied over those inputs, as further discussed in Section 7.1.3.

7.1.2 Data Representation

To handle probabilistic data within the analysis process, the CE was extended by appropriate distribution data types as well as basic operations for accessing the distribution information.

Representation of Probabilistic Data

The distributions of variables are internally represented either symbolically or as histograms. Those data are processed (consumed and produced) by the provided operators at runtime and can be stored in dedicated database tables. A unique identifier `v_id` is assigned for each variable and associated with a distribution representation. Symbolically represented distributions are represented by means of parameter values in a symbolic distribution table, which associates each `v_id` with a distribution family and corresponding parameters. For histogram representations, a relation `HISTINFO(v_id,v_lower,v_upper,beta,p_id)` stores information about the lower and upper support of each histogram along with the number of bins, β. A relation `HISTOGRAM(v_id,b,w)` stores the actual distribution information in the form of β frequency values w_i associated with the bins b_i. Optionally, the `p_id` column of `HISTINFO` contains a key value that refers to a "parent" histogram v_id in the same table. Thus, modifications can be stored in the form of delta histograms Δ^H and related to their corresponding base histograms through the `p_id`. At query time, it is possible either to access only the delta histograms (e.g., in order to evaluate impact predicates ϕ_i), or to combine the deltas with their base (parent) histogram data. Within a CV, probabilistic attribute data are referenced through values of integer-valued input and output columns, which hold the variable identifiers (`v_ids`). The distribution data associated with a set of `v_ids` is loaded at runtime when the operator that computes the output of the current CV is executed.

Physical Storage Model

On the physical level, SAP HANA supports both a column-based and row-based storage model. Due to the focus of this thesis on data analysis tasks, the column-store was chosen as the basic storage model for the prototype. First, the separate storage of individual columns (attributes) rather than rows (tuples) is beneficial for analytical queries, where operations often need to access values from a small number of columns per row, rather than accessing entire rows. This is true both for standard OLAP operators [ABH09, Pla09, JLF10] as well as for the techniques and operations presented in this thesis, particularly those that process large amounts of raw fact data, such as $\mathcal{D}$ and $\mathcal{E}$.

Moreover, the column-store model enables a flexible addition of columns that hold intermediate or final result data to the output of a CV, which can be (re-)used by further CVs in a CS. Finally, for operators that process multiple input columns, yet process those independently, the columns can be processed in a parallel fashion, thus opening opportunities for optimization.

7.1.3 Operator Implementation, Composition, and Execution

As stated above, an individual CV can be considered analogous to a database view, and can either implement access to selected columns of a base table, or an operation applied over provided input columns of one or several input CVs or DSs. This enables users to apply both standard OLAP analysis functionality (e.g., filtering or aggregation of deterministic fact data) as well as using custom (analytical) operators which have been previously supplied by a consultant or an expert user.

The operators discussed in Sections 3.3, 4.3, and 4.4 are thus implemented as C++ custom operators. Each such operator takes a collection of columns (which are accessed from one or several *InternalTable*s) as input and produces a new collection of columns (another *InternalTable*) as output. To apply a specific operator, one defines a new CV, in which the desired operator is referenced through its associated operator identifier and the required parameter values are provided through a key-value map. Further, one specifies the input CVs and the columns that are consumed and produced by the operator. At query processing time, the operator accesses its internal input table(s), from which it then retrieves all relevant input columns. Then, the operator iterates over all values of the column(s) according to its specific implementation. One or several result columns are constructed and added to the internal output table.

The output columns produced by the analytical operators implemented within this thesis contain either a set of result variable references, or deterministic values, reflecting result probabilities, quantile values, or expected values that are in turn computed from probabilistic input data.

Composition of Operators As stated before, a CS is constructed by specifying and composing several CVs. The topmost CV of a scenario along with its ancestors form a directed acyclic graph, which is stored as a CS. Each CV can have several input CVs (or DSs) and one output CV. A CS is represented and stored by means of a JSON data object, i.e., consistent with a dedicated JSON Calculation Scenario schema. A thus specified CS is created at the database through an SQL statement and associated with a unique name under which it can be queried. A previously created scenario and its contained CVs can be queried like an ordinary database view, and their outputs can serve as input to further CVs that may be specified in other CSs. The consistency of a CS is checked at definition time based on the specification of the input and output tables and columns of successive CVs.

Parallelization The use of custom operators largely prevents the built-in automatic parallelization of the HANA Calculation Engine. Thus, one needs to rely on different schemes for parallelizing their execution "manually" to increase the runtime efficiency by scaling up to available resources. To this end, one can use either intra-view or inter-view parallelization, depending on the characteristics of the application case. While intra-view parallelization is implemented through threaded execution of custom operators within a single CV, inter-view parallelization relies on several CVs being computed in parallel, each one processing one of several partitions of the data. Since the partitioning required by the inter-view approach must be specified and performed explicitly before the actual calculation, it is not applicable in the context of ad-hoc analysis processes. Thus, the intra-view parallelization was chosen as the more appropriate approach. This parallelization approach was prototypically implemented and evaluated for the correlation handling operators ($\mathcal{C}$ and $\mathcal{E}$).

7.2 Evaluation Objectives

The next sections present the evaluation of the functionality discussed in Chapters 3 through 5. The evaluation addresses a variety of goals and is organized as follows.

First, we discuss the application and composition of the provided operators such as to enable the implementation of the use cases presented in Section 1.2 and fleshed out during the course of the thesis. Then, Sections 7.4 and 7.5 focus on the evaluation of the runtime efficiency for the operator functionality presented in Chapters 3 and 4, relating the experimental results to the analytically derived computational complexities. Specific focus is put on a thorough evaluation of the functionality for correlation handling. The subsequent section then addresses the runtime performance of the recomputation approaches discussed in Chapter 5. Finally, the performance of *HANA Ext* is compared to the MC-based probabilistic database systems in Section 7.7. There, we also discuss an alternative, sample-based implementation of the correlation introduction operator $\mathcal{C}$ and discuss its drawback with respect to the ACR-based approach of this thesis.

7.3 Use Case Implementations

This section describes an implementation of the use cases discussed in Section 1 and further extended throughout the thesis.

7.3.1 Implementation of Use Case 1

Use Case 1 addressed the analysis and prediction of possible sales scenarios. The tasks of Use Case 1 cover both the application of basic operators such as the derivation of distributions from fact data ($\mathcal{D}(\dot{X})$) and the calculation of expectations ($\mathbb{E}(x)$), as well as the application of the temporal aggregation using $\mathcal{A}^{\underline{T}}$. The data schema of the underlying sales dataset is depicted in Figure 7.1. It contains information about historic sales figures (margin, quantity, revenue), and their associated dimensional data (e.g., dimensions holding attributes that describe the sold product, the store and the week of the sale).

Figure 7.2 illustrates the implementation Use Case 1, Task A, as visualized by a user interface included with the HANA database administration system. The CS uses the DS SALES, which provides the historic sales fact table and the DS WEEKS, which provides the temporal dimension data. The CV SALES_IN_2011 selects all entries joins the temporal dimension data and those sales that are associated with the relevant time range (2011) and product (i.e., the product which serves as reference product for the newly introduced product), and the distribution of monthly revenues per store is calculated as the output of the CV DERIVE_REVENUE. Then, the expected revenues are calculated simply by computing the expected value of a linear function of the derived distributions. For example, the user may assume an increase of 1%, 5%, and 8% as a weak, medium, and strong effect of the marketing campaign on the revenue of the new product; the results of EXP_REVENUE1, EXP_REVENUE2, and EXP_REVENUE3 are then computed as $\mathbb{E}(1.01 \cdot x)$, $\mathbb{E}(1.05 \cdot x)$, and $\mathbb{E}(1.08 \cdot x)$, respectively. Note that this basic implementation assumes a perfect linear dependency structure. Alternatively, the influence of the

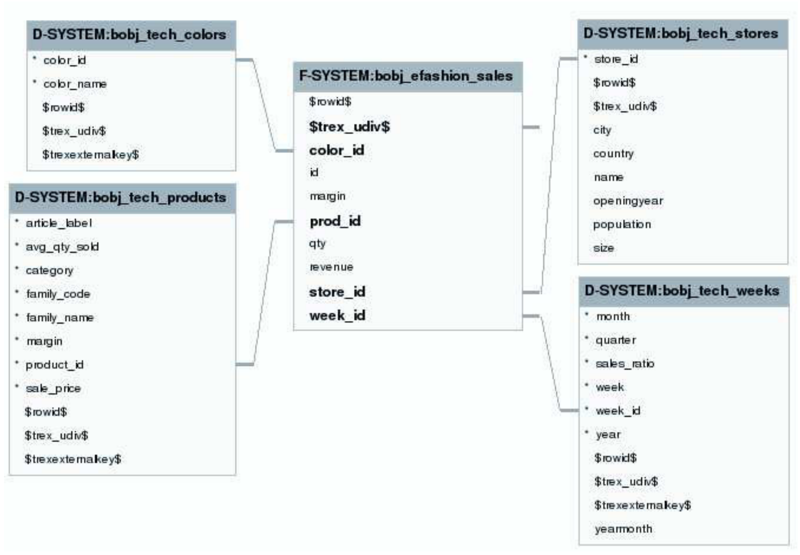


Figure 7.1: Star schema underlying Use Case 1

marketing campaign could be analyzed using an appropriate ACR, as discussed in the context of Use Case 2 below.

In Task B of Use Case 1, the prospective revenue in the first half of 2012 shall be computed for a new product based on the forecast revenue of a reference product. The graph that results from the implementation of this task is shown in Figure 7.3. The CS uses the first revenue expectation scenario from the previous task as input data, represented through the CV `PROSPECTED`, which associates a reference product ID with the previously computed expected monthly reference revenue. A second input is provided by the DS `RELEASES` which holds the `prod_id` of the newly released product(s), the `ref_id` of the reference product, and variables `start_id` and `dur_id` representing the uncertain start time and duration of the sales period of each product. After joining the two tables on `ref_id=prod_id`, the temporal aggregation is applied in the CV `Q1Q2REVENUE` over the set of temporally indeterministic sales events to calculate the aggregated `total_revenue` for each `prod_id`. That is, the operator $\mathcal{A}^{\underline{\mathcal{T}}}$ is applied in order to compute the expected sales during the first two quarters of the year 2012 under indeterminate launch times of the newly released product.

7.3.2 Implementation of Use Case 2

With special focus on the relevance on correlation in business data, Use Case 2 addressed the analysis of risks related to jointly occurring events in the insurance domain. To support such analyses, Chapter 4 discussed the functionalities for both extracting and introducing correlation information (using $\mathcal{E}$ and $\mathcal{C}$), as well as the computation of joint measures over thus correlated data (using $\mathcal{T}^m$ and $\mathcal{M}$). Those functionalities are required to implement this use case.

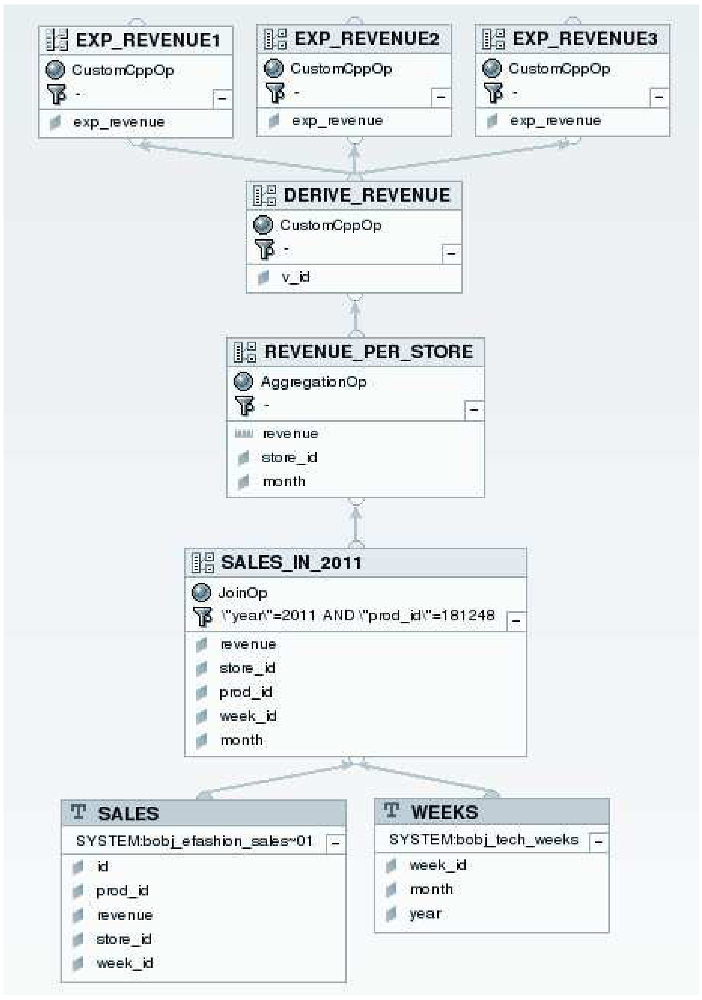


Figure 7.2: CS graph for Use Case 1,Task A

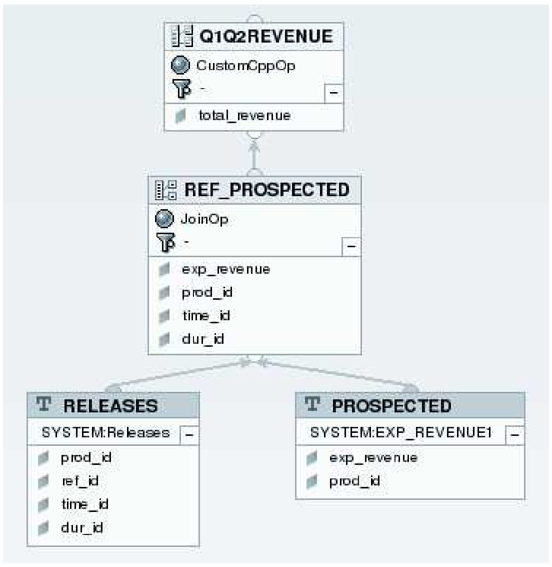


Figure 7.3: CS graph for Use Case 1,Task B

Task A In Task A of Use Case 2, the user wants to evaluate the risk of high flood damages *under the assumption* of different weather conditions, say regarding rainfall. The task can be implemented by applying $\mathcal{C}$ over the marginal input distributions associated with weather and flood forecasts using a selected ACR. Figure 7.4 shows the visualization of the corresponding CS graph. The input data is provided by the DSs `WEATHER_FC` and `FLOOD_FC`, which hold variables associated with forecasts for the future rainfall strength and flood-related insurance claims. The user first assumes a correlation between rainfall and flood damages, which is introduced in the CV `WEATHER_FLOOD_JOINT`. The CV `COPULA_H,d`, reflects the ACR selected from the ACR store (`COPULAS`) based on the parameters determining the structure `H` and strength `d` of the desired correlation. Based on the result data of `WEATHER_FLOOD_JOINT`, different marginal distributions (`FLOOD_MRG1`,`FLOOD_MRG2`) are then computed, from which the user finally calculates the tail probabilities (risks) that the flood damages will exceed, e.g., $\$15bn$ (CVs `TAIL_FLOOD1`,`TAIL_FLOOD2`).

Task B In Task B, the goal is to evaluate risks associated with the *joint occurrence* of insurance events under different correlations. The first case, Task B.1, addresses the evaluation of catastrophe-related claims based on an assumed correlation. This task is implemented by applying $\mathcal{C}$ with (possibly several pairs of) x_f and x_h as marginal input distributions and using different ACRs, e.g., $\overline{C}_{T(1),0.8}$, as shown in Listing 7.1. In a second step, the joint probability $p^u_{15\,bn,15\,bn}$ is computed over the resulting joint distribution $x_{f,h}$ by applying $\mathcal{T}^m$.

Listing 7.1: Query for Task B.1

```
WITH jointDfh AS
  (SELECT C(x1,x2,acr)
```

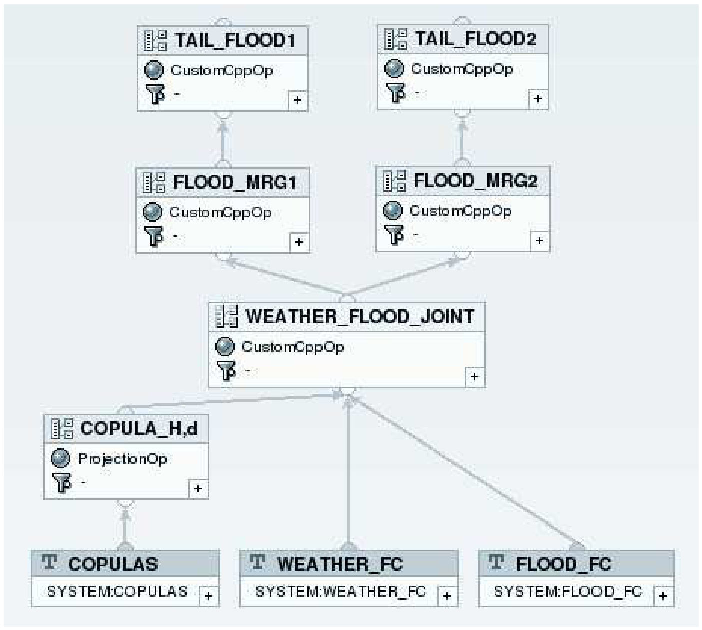


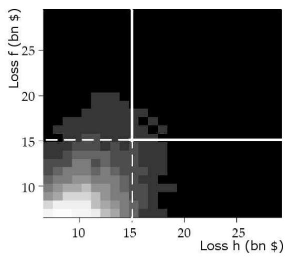
Figure 7.4: CS graph for Use Case 2,Task A

```
  FROM Distribution x1, Distribution x2, Copulas acr
  WHERE x1.id = 'xf' AND x2.id = 'xh'
  AND acr.H = 'T(1)' AND acr.d = 0.7)
SELECT Tᵘ(jointDfh, 15, 15)
```

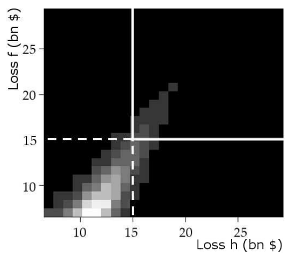
In Figures 4.3(c) and 4.3(d) (see page 63, Section 4.2.3), the samples for which the prospective loss both from flood and hurricane claims exceeds the threshold of $\$15\,bn$. A comparison of those figures shows that in the case of $\overline{C}_{G,0.8}$ a smaller mass falls into the extreme region than in the case of $\overline{C}_{T(1),0.8}$. This is confirmed through the query result visualization in Figure 7.5. The joint probability $p^{u}_{15\,bn,15\,bn}$ is calculated by summing up the bin density values over the marked regions of the histograms. The higher tail dependency introduced through the ACR $\overline{C}_{T(1),0.8}$ (adding up to a risk of 1.3%) as opposed to $\overline{C}_{G,0.8}$ (0.7%) can be observed in the visualized distributions. For an insurer, evaluating such alternatives is essential: Even small underestimations of risk may lead to an existential threat, as it implies premium reserves that are too low to cover claims in a "worst case" (such as a total of $\$135\,bn$ in property damages caused by Hurricane Katrina alone[1]).

In Task B.2, the user considers a different class of insurance events—burglary and vehicle theft—, from which he first wants to extract a correlation structure, in order to then apply it to *forecasts* of burglary and vehicle theft, b_{fc} and v_{fc}. Listing 7.2 illustrates the query that implements this task. First, the user needs to derive an ACR from the samples for burglary and vehicle theft claims over a relevant time frame. He does so by applying $\mathcal{E}$, e.g., with a

[1]Swiss Re sigma, catastrophe report 2005



(a) Using $\overline{C}_{G,0.8}$



(b) Using $\overline{C}_{T(1),0.8}$

Figure 7.5: Result histograms $\overline{D}_{f,h}$ (right upper region)

desired result ACR granularity of $\alpha = 10$. Then, the extracted ACR is applied to correlate the distributions associated with b_{fc} and v_{fc}.

Listing 7.2: Query for Task B.2

```
CREATE VIEW Claims_Per_Week AS
  SELECT SUM(bClaim), SUM(vClaim)
  FROM V_Claims v, B_Claims b WHERE b.week_id = v.week_id
  GROUP BY week_id

WITH Emp_BV AS (
    SELECT E(Claims_Per_Week, 10)
    FROM Claims_Per_Week)
  SELECT C(B_FC,V_FC, Emp_BV)
    FROM B_FC, V_FC, Emp_BV
    WHERE B_FC.week_id = V_FC.week_id
```

Figure 7.6 shows the visualization of the CS graph resulting for the implementation of Task B.2. In contrast to the implementation shown in Figure 7.4, an additional CV `EMP_BV` implements the extraction of the empirical copula between weekly burglary and theft claims (which is first selected and aggregated from fine-grained claim data on a weekly level). The result of the CV `EMP_BV` is then used as input to the CV `BV_JOINT`. In the `BV_JOINT` view, the extracted ACR serves to transfer the historical correlation to (possibly several) pairs of forecast values b_{fc} and v_{fc}, which are provided by DS `B_FC` and DS `V_FC`.

Alternatively to investigating the correlation during the complete time frame (1990-2009) of the underlying fact data, as shown in the listing, the user can also evaluate the correlation within different time windows, by first selecting data from within those time windows, and then deriving different ACRs over each of the selected data sets in turn.

As an example, consider the case of two 10-year windows $T1 = [1990, 1999]$ and $T2 = [2000, 2009]$. Figures 7.7(a), 7.7(b) and 7.7(c) show the ACRs $\overline{C}_T^{10}$, $\overline{C}_{T1}^{10}$ and $\overline{C}_{T2}^{10}$ derived from all samples, and from the samples during $T1$ and $T2$, respectively. In this case, the more recent data show a different correlation structure, with a far lower dependency in the upper tails (i.e., a lower density in the upper right quadrant of Figure 7.7(c)). For introducing correlation to the forecast data, the user might indeed apply $\overline{C}_{T2}^{10}$, since it represents the more recent

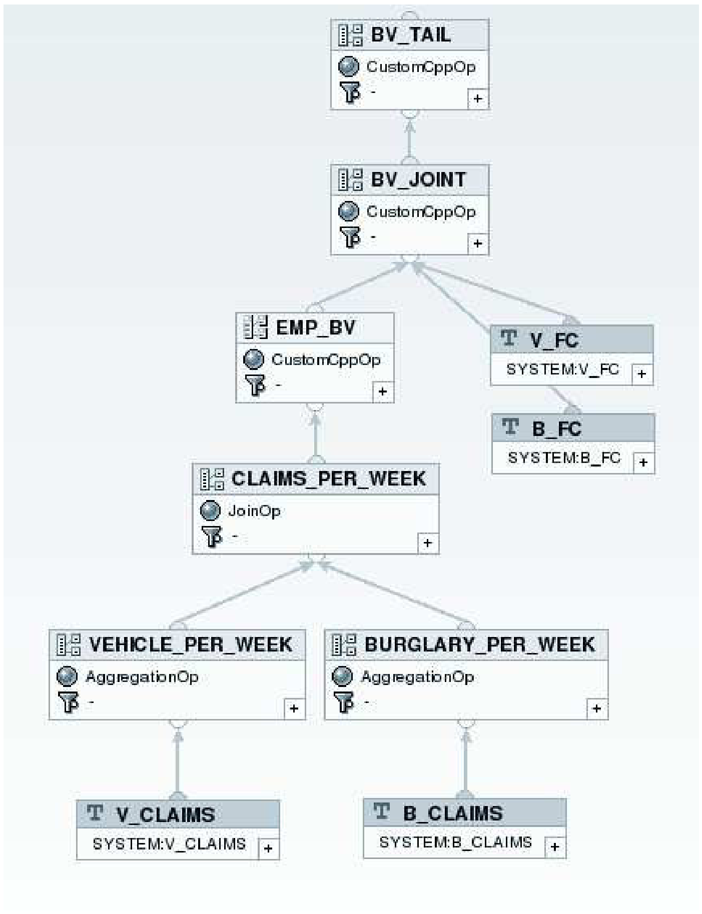
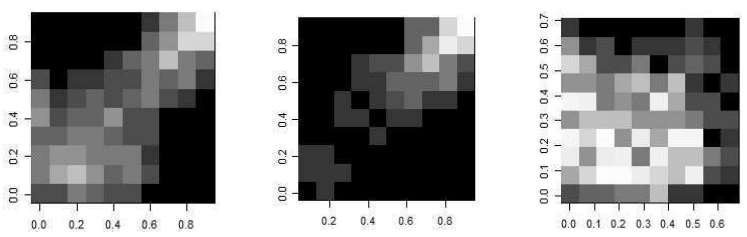


Figure 7.6: CS graph for Use Case 2,Task B.2

correlation structure, which is more likely to hold for the future development. In this case, a subsequent computation of the joint tail probability results in a risk of below 1%. Noteworthy, the risk would have been estimated much higher (12%) when applying the ACR $\overline{C}_T^{10}$, and again even higher (23%) using $\overline{C}_{T1}^{10}$. In this case, the overestimation of risk could have lead the user to opt for an overly cautious strategy, e.g., regarding the insurance premium reserve, and therefore, loss of potential profit (chances) for the insurer.

7.3.3 Summary

This section discussed exemplary implementations of the use cases which have been discussed throughout the thesis, and showed how the provided operators are applied to fulfill the required



(a) $\overline{C}_T^{10}$ (b) $\overline{C}_{T1}^{10}$ (c) $\overline{C}_{T2}^{10}$

Figure 7.7: ACRs derived from all samples and two 10-year time windows, respectively

functionality. Particularly, the beneficial aspects of the provided correlation handling functionality and their flexible application for Use Case 2 were illustrated. By means of exemplary CS graphs, the composition of analysis processes as in the *HANA Ext* implementation was demonstrated.

The next section turns to the evaluation of the runtime performance of *HANA Ext*, regarding both the application of individual operators and the recomputation of analysis queries.

7.4 Univariate Analysis and Aggregation Operations

In this section, the runtimes of the univariate operators discussed in Chapter 3 are evaluated.

7.4.1 Setting

All of the following tests were executed on a server equipped with a Quadcore CPU 2.0GHz and 32GB RAM running SLES 11.0. The underlying dataset is the TPC-H benchmark data, which was generated with a scaling factor of 1 (resulting in a total database size of 1GB, or a corresponding number of $6m$ entries in the `lineitem` fact table). The $\mathcal{D}$ operator is used to derive distributions over the data in the `lineitem` tables for each associated `l_suppkey`, `o_custkey`, or `l_partkey`. This results in a total of $10k$, $100k$ or $200k$ distributions, respectively. Those distribution data are then used as input data in the subsequent evaluation cases. For example, the aggregation operators are executed over a set of $10k$, $100k$, and $200k$ random variables associated with a derived distribution each. Further, for evaluating $\mathcal{A}^{\underline{T}}$, an uncertain `l_shipdate` and an uncertain `l_shipduration` is assigned to each of the facts in `lineitem`, each distributed uniformly within $[0, 100]$[2].

[2]The distribution was chosen for simplicity and without restriction of generality. The concrete shape of the individual distributions is irrelevant in the context of the evaluation of $\mathcal{A}^{\underline{T}}$ provided it is representable by one of the supported distribution families or by a histogram representation.

7.4.2 Runtimes

This section presents the runtime results for the basic operators for deriving and analyzing univariate distributed data, i.e., $\mathcal{D}$, σ^τ, $\mathcal{A}_{SUM}$, $\mathcal{A}_{MIN/MAX}$, and $\mathcal{A}^{\underline{T}}$.

Deriving Distributions

For the evaluation of $\mathcal{D}$, we consider the derivation of histogram-based and symbolic representations over the `l_extendedprice` attribute of the `lineitem` table. Figure 7.8 shows the resulting runtimes when grouping the data by `l_partkey`, resulting in a total of $200k$ distributions. The target histogram granularity is set to $\beta = 10, 20, 50$, or 100, respectively. For deriving symbolic representations of the underlying distributions, we fit a uniform, a Gaussian, or a Gamma distribution.

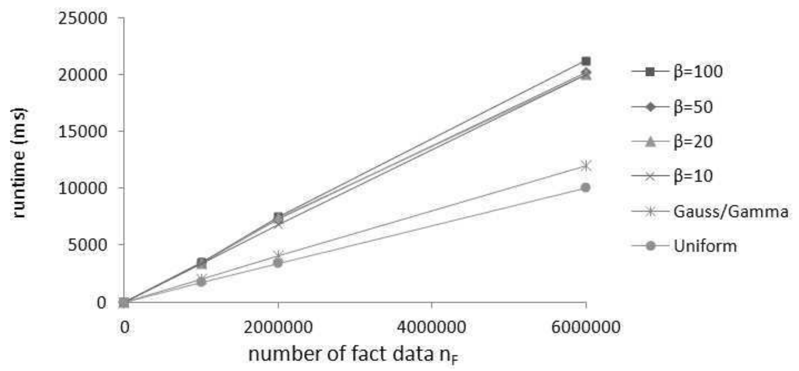


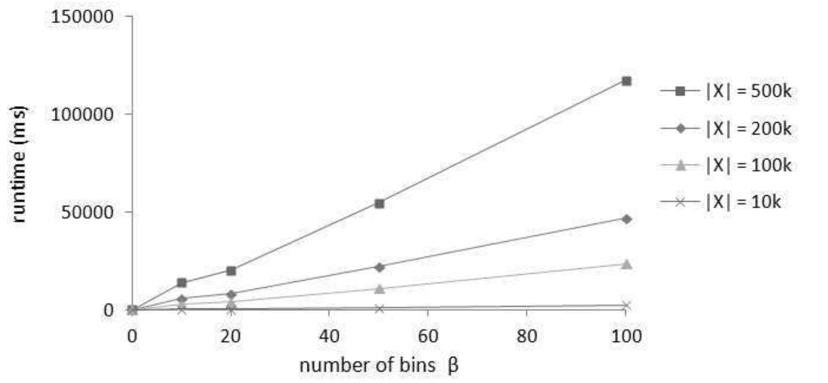
Figure 7.8: Runtimes of $\mathcal{D}$ for varied datasets and approximation granularities

As shown in Figure 7.8, the runtimes for all evaluated cases increase linearly with the size n_F of the input fact data, as expected based on the complexity of $\mathcal{O}(n_F)$. For the construction of histograms, the increase in β causes a slight relative increase in runtimes, even though the bin allocation costs as such are constant due to the statically determined bin bounds. This effect is due to increasing costs for writing the resulting histograms to the output *InternalTable* of the corresponding CV.

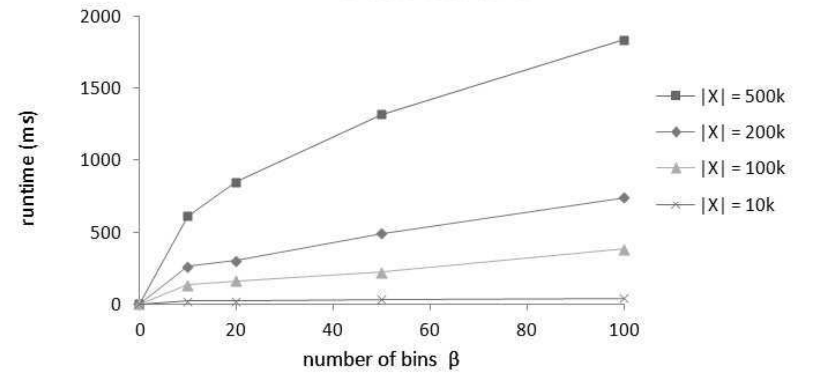
Probabilistic Selection

The probabilistic threshold selection operator, σ^τ, is applied to the previously derived sets of variables, X. The associated distributions are being represented in different granularities, i.e., $\beta = 10, 20, 50$, and 100. As shown in Figure 7.9(a), the overall runtimes for σ^τ are linear in β. This behavior (which is inconsistent with the expected complexity of $\mathcal{O}(|X| \cdot \log_2 \beta)$) is due to the dominating effect of the loading times, which are linear in β.

In contrast, Figure 7.9(b) shows the *calculation* times of the same operator, i.e., without the times required for loading and constructing the distribution information. There, the expected logarithmic behavior in β can be observed.



(a) With loading



(b) Without loading

Figure 7.9: Runtimes of σ^τ for varied datasets and approximation granularities

Aggregation

The computation of $\mathcal{A}_{SUM}$ and $\mathcal{A}_{MIN/MAX}$ over X is again evaluated under varying approximation granularities β of the histogram representations associated with each $x \in X$. The results for the application of $\mathcal{A}_{SUM}$ for various sizes of the processed data set X are shown in Figure 7.10.

The runtimes resulting for the execution of $\mathcal{A}_{MIN/MAX}$ are almost equal to those for $\mathcal{A}_{SUM}$, regardless of the difference in their associated complexities. This behavior is again due to the fact that the costs for loading and constructing the distribution representations dominate the overall runtimes. In contrast, Figures 7.11(a) and 7.11(b) show the pure costs for calculating the results of $\mathcal{A}_{SUM}$ and $\mathcal{A}_{MIN/MAX}$, respectively. There, one can observe that the runtimes are linear in β for $\mathcal{A}_{SUM}$ and logarithmic in β for $\mathcal{A}_{MIN/MAX}$, which is consistent with the complexities of $\mathcal{O}(|X| \cdot \beta)$ and $\mathcal{O}(|X| \cdot \log_2 \beta)$, respectively.

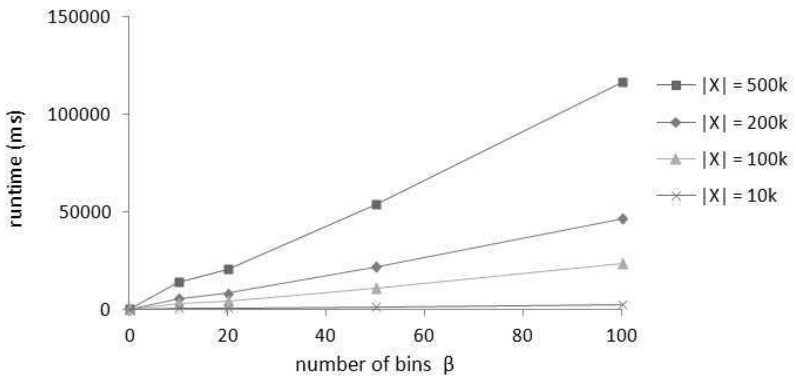
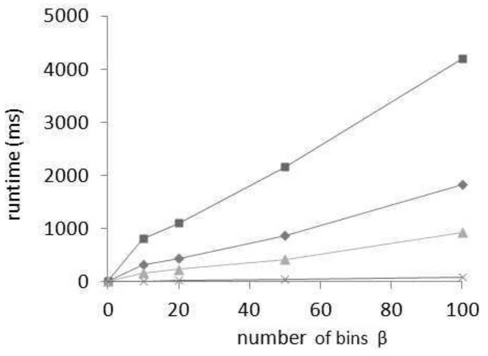
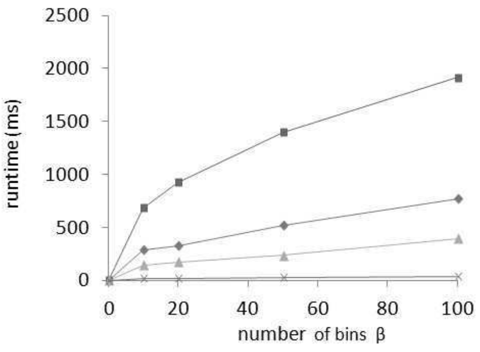


Figure 7.10: Runtimes of $\mathcal{A}_{SUM}$ for varied datasets and approximation granularities



(a) $\mathcal{A}_{SUM}$



(b) $\mathcal{A}_{MIN/MAX}$

Figure 7.11: Calculation times of $\mathcal{A}_{SUM}$ and $\mathcal{A}_{MIN/MAX}$

Temporal Aggregation

The temporal aggregation operator $\mathcal{A}^{\underline{T}}$, is evaluated under different granularities β_{t_i} and β_{d_i} of the distributions associated with the start times and durations of a set of events E. Further, the overlap fraction f_ϕ is varied. As input data, a set of $10k$ indeterminate line item shipping events is assumed. As note above, each such event is associated with an uncertain start time and duration uniformly distributed over $[0, 100]$. Each time point or duration is represented by a corresponding histogram with $\beta_{t_i} = \beta_{d_i} = \beta = 10, 20, 50$, or 100 bins.

In the first considered test case, β_{t_i} and β_{d_i} are varied. The portion f_ϕ of overlapping occurrence intervals 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 for various sizes of the aggregated data sets are displayed in Figure 7.12. There, one can observe a quadratic increase in β.This behaviour is consistent with the complexity of $\mathcal{O}(|E| \cdot n_I \cdot f^\phi)$ with $n_I = \beta_{d_i} \cdot \beta_{t_i}$ and $\beta_{t_i} = \beta_{d_i} = \beta$.

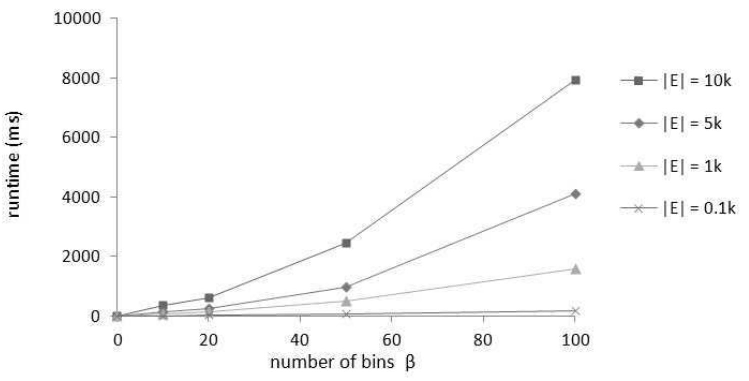


Figure 7.12: Runtimes of $\mathcal{A}^{\underline{T}}$ for varying β_{t_i}, β_{d_i}

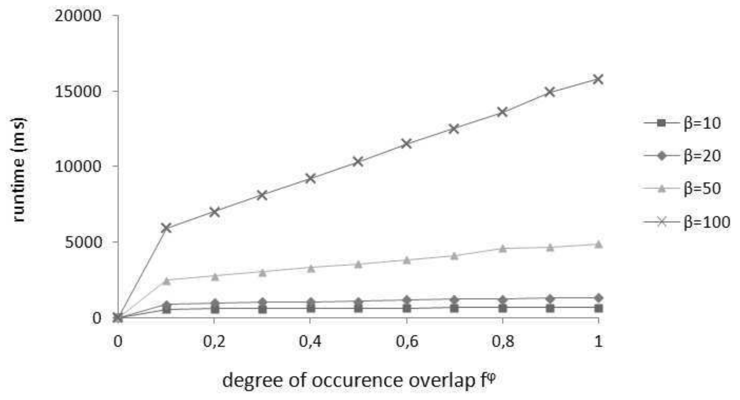


Figure 7.13: Runtimes of $\mathcal{A}^{\underline{T}}$ for varying fraction of overlaps f_ϕ from 0.0 to 1.0

In a further experiment, the fraction f_ϕ of the potential occurrence intervals I_{ipq} that do overlap $\underline{T}$ is varied. To vary the portion f_ϕ of overlapping I_{ipq} from 0.0 to 1.0, t_i and $\underline{T}$ are kept constant and $\mathcal{A}^{\underline{T}}$ is calculated for varying distributions of d_i. The resulting runtimes are shown in Figure 7.13. Again, a linear increase of runtimes can generally be observed, which is in line with the analysis results. An exception occurs for the smallest degree of overlap, $f_\phi = 1/10$, where we observe an initially stronger increase. This is due to the fact that for $f_\phi = 0.0$, all potential occurrence intervals of all events lie outside $\underline{T}$, which can be determined by solely accessing $\underline{t_i}$ and $\overline{d_i}$. For any $f_\phi > 0.0$, the test whether an interval overlaps $\underline{T}$ requires that we load the representations of t_i and d_i, which in turn implies a higher initial cost.

7.5 Correlation Handling and Multivariate Analysis

This section considers the performance of the approaches for ACR extraction, correlation introduction, and analysis over the resulting joint distributions. The experiments address both the efficiency and the accuracy of the discussed functionality.

7.5.1 Setting

To demonstrate the flexible applicability of the proposed operators for correlation handling, the application of ACRs is being evaluated under various correlation parameter settings. Further, the ACR-based correlation results are compared to those obtained by the sample-based approach considering the runtimes and result accuracy. Since the basic approach for correlation introduction ($\mathcal{C}^{basic}$, see Section 4.4.1) often results in large discretization errors, we only consider the application of the uniform spread approach (applying $\mathcal{C}^{resample}$ and $\mathcal{C}^{reverse}$, see Section 4.4.2). Those approaches are more expensive than using $\mathcal{C}^{basic}$, yet at the benefit of a considerably higher accuracy.

Use of Precomputed ACRs

In the following experiments, the operators $\mathcal{C}^{resample}$, $\mathcal{C}^{reverse}$, $\mathcal{E}$, $\mathcal{T}$, and $\mathcal{M}$, which are implemented as C++ operators as discussed in Section 7.1, are evaluated. For the ACR-based test cases, a set of ACRs was precomputed. To this end, the R copula package was used [KY10]. When using $n_{ACR} = 100k$ samples for the copula pre-computation, the construction and storage of an ACR takes between 0.5 to 1 minutes depending on the type of underlying generation function and the granularity α of the produced ACR. It must be noted again that those time requirements are irrelevant for the application of copulas at runtime. The following table lists the space required for the storage of the precomputed ACRs (in uncompressed form). It can be seen that especially low-granular ACRs induce very low memory costs. Thus, a large variety of ACRs can be precomputed and kept in memory for faster access at execution time.

α	10	20	40	100
size (kB)	14	29	56	181

Test Cases

Different marginal distributions (Gaussian, Gamma, and T(1)) and various copulas (Gaussian (G), T(1), and Clayton (C)) are used as input data in the experiments. This way, the query result accuracy can be evaluated for different variations of correlated data as well as for different correlation structures. At the same time, this variable parameterization stresses the flexibility of our approach.

Baseline distribution histograms $x_{1,2}^B$ were constructed from $100k$ samples for a highly accurate representation. They serve as a basis for comparing the accuracy of the resulting joint distributions, as well as the times required for their computation. In particular, six variants of this baseline were computed, using copulas $C_{G,0.4}$, $C_{G,0.8}$, $C_{T(1),0.4}$, $C_{T(1),0.8}$, $C_{C,0.4}$, and $C_{C,0.8}$. For each of those variants the sample-based approach was then evaluated using a smaller sample size of $n_S = 5k$, $10k$, and $40k$ copula samples, respectively. Similarly, the

ACR-based results were computed using ACRs that approximate the six copulas with $\alpha = 10$, 20, and 40 bins.

Apart from the operator $\mathcal{C}$ (in its variants $\mathcal{C}^{resample}$ and $\mathcal{C}^{reverse}$), the runtimes for the derivation of empirical copulas from historic data using $\mathcal{E}$ were also measured. The results are presented in Section 7.5.3.

7.5.2 Accuracy

For evaluating the accuracy of the resulting joint distributions, the root mean squared error between the evaluation results and the baseline results is computed. That is, the densities $f^B_{k,l}(k = 1, ..., \beta_1, l = 1, ..., \beta_2)$ in $x^B_{1,2}$ are considered as the expected (baseline) values and the densities $f^E_{k,l}$ of the evaluation results $x^E_{1,2}$ are considered as the observed values. Then, the error is computed as the root mean squared error

$$e = \sqrt{(\sum_{k,l} (f^B_{k,l} - f^E_{k,l})^2)/N}$$

over all $N = \beta_1 \cdot \beta_2$ bins of the result histogram.

Figure 7.14 shows the errors obtained for each of the sample-based and ACR-based evaluation variants when correlating the Gamma distribution x_h and a standard normal distribution x_G, represented through $\beta_h = \beta_G = 40$ bins each. As the most accurate representations, the usage of copulas with $40k$ samples clearly dominates all other results. However, the error statistics for the ACR variants also indicate high result accuracies. In particular, very accurate results (around 0.5% error) are obtained for the cases of lower correlation (i.e., $d = 0.4$) even with a low ACR granularity α. One can see that the error increases slightly with larger values of d (i.e., $d = 0.8$). This effect is due to the larger approximation error in the copula regions corresponding to the joint tails and can be mitigated through a more accurate representation of those regions as described in Section 4.3.

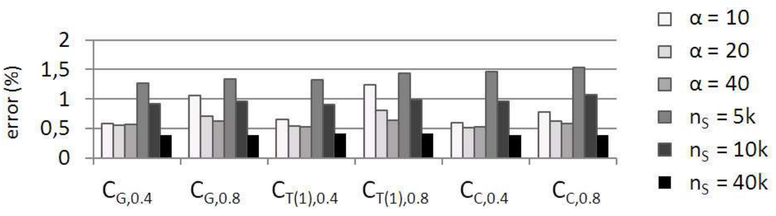


Figure 7.14: Accuracy of $\mathcal{C}$ for various copulas $C_{H,d}$ and ACRs $\overline{C}^{\alpha}_{H,d}, H \in \{Gauss, T(1), Clayton\}, d \in \{0.4, 0.8\}$

A further experiment investigates the accuracies of the joint distributions that are obtained when correlating *various marginal distributions*. Three ACR-based and three sample-based correlation cases are compared. For each of them, the marginal distributions x_1 and x_2 are chosen to be equal, both following either a Gaussian x_G ($\mu = 0, \sigma = 1$), a (heavy-tailed) T-Student distribution x_T with one degree of freedom, or the (light-tailed) Gamma distributions

x_f or x_h from Use Case 2, respectively. The results are shown in Figure 7.15. One can see that, similar to the accuracy results reported for different correlation structures above, $\mathcal{C}$ computes joint distributions which deviate from the true joint distribution only within a small error bound of 1%. Those results show that the correlation introduction performs highly accurately for different types of input (marginal) distributions.

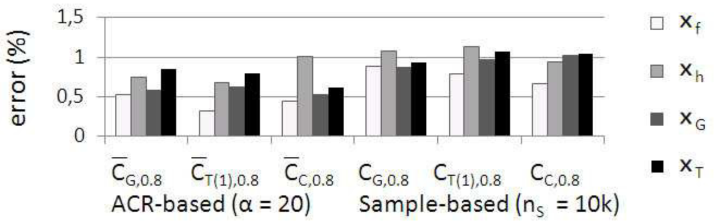


Figure 7.15: Accuracy of $\mathcal{C}$ using different ACRs $\overline{C}^{20}_{H,0.8}, H \in \{Gauss, T(1), Clayton\}$ and copulas $C_{H,0.8}$, $n_s = 10k$ to correlate varying marginals x_1, x_2

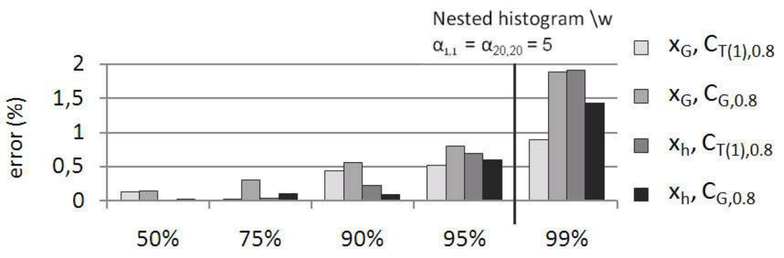


Figure 7.16: Accuracy of resulting tail probabilities

However, as shown in the scenario, in many queries over correlated data the focus lies particularly on the tails of a joint distribution rather than on the whole distribution. Therefore, instead of the overall accuracy of $x_{1,2}$, the accuracy of a selected part (e.g., the right upper corner as marked in Figures 4.3(c) and 4.3(d)) is of specific interest. To this end the ACRs $\overline{C}^{20}_{T(1),0.8}$ and $\overline{C}^{20}_{G,0.8}$ are applied and the accuracy of the joint tail probabilities $p^u_{t_1,t_2}$ over the resulting joint distributions is evaluated. The correlated marginals x_1 and x_2, represented through $\beta_1 = \beta_2 = 40$ bins, follow either the Gaussian x_G or the Gamma distribution x_h. The thresholds t_1 and t_2 are set so that they correspond with different percentiles $(50\%, 75\%, 90\%, 95\%, 99\%)$ of each of the marginals. The results displayed in Figure 7.16 show that for all cases but the last one (i.e., for $t_1 = inv_1(0.99)$ and $t_2 = inv_2(0.99)$), the results are highly accurate (with an error below 1%). The higher inaccuracy in the very upper tails of the marginals is due to the fact that the corresponding region of the copula is not represented accurately enough. To avoid large result errors, one can alternatively apply ACRs where the copula region covered by the ACR bins $b_{1,1}$ and $b_{20,20}$ is further partitioned by a nested histogram, as described in Section 4.3.4. For the experiment, the granularity was set to $\alpha_{1,1} = \alpha_{20,20} = 5$, resulting in a representation granularity of $0.05/5 = 0.01$ for the two extreme joint tail bins. The results displayed in Figure 7.16 for the 99^{th} percentile shows that the joint tail probability can now be computed very accurately, with below 2% relative error for all the test cases.

7.5.3 Runtimes

Another set of experiments was conducted to evaluate the runtimes of the $\mathcal{C}$ operation, as well as the runtimes of the empirical ACR derivation using $\mathcal{E}$.

Empirical ACR Derivation As the construction of empirical ACRs takes place at query time (except for the case where the user accesses previously stored empirically derived results), it is crucial that the runtimes of $\mathcal{E}$ are sufficiently low. For the evaluation, copulas that reflect the correlation of the `l_extendedprice` and `l_quantity` attributes of the TPC-H `lineitem` table were constructed. Different sizes of sample sets were used, containing $|F| = 10k$, $50k, 100k, 500k$, and $1000k$ facts for deriving the ACR. A parallelized version of $\mathcal{E}$ was implemented as described in Section 4.5.2 to speed up the ACR extraction. Figure 7.17 illustrates the required times for extracting ACRs with $\alpha = 40$ bins for a single-threaded execution as well as different multi-threaded settings.

The results show that the parallelization allows runtimes well below $1s$ even for large $|F|$. Runtimes do not scale linearly with the number of threads due to the costs for merging the t partial marginal distributions, as well as building the final ACR. Still, the resulting runtimes allow for an ad-hoc derivation of empirical ACRs and thus, the flexible analysis of correlation structures existing in data.

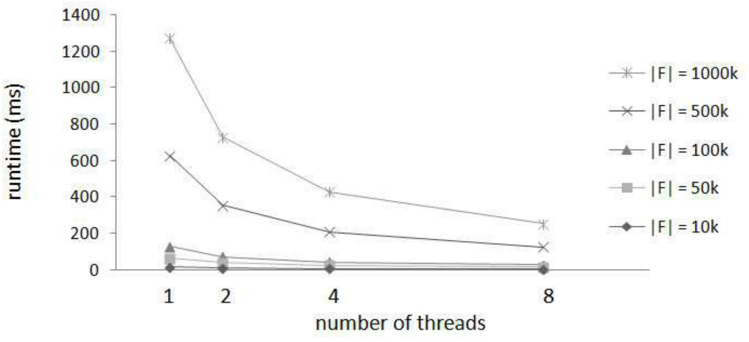


Figure 7.17: Runtimes for empirical copula derivation

Note that any result of $\mathcal{E}$ can serve as input to $\mathcal{C}$ just like the precomputed ACRs, which were used in the experiments of the previous section. The accuracy results presented in Section 7.5.2 therefore apply similarly to the use of empirical copulas $\overline{C}^{\alpha}_{F}$.

Introducing correlation Given an empirically derived ACR or a selected precomputed ACR, the goal is to enable fast correlation processing using $\mathcal{C}$, as well as efficient processing of subsequent analyses using $\mathcal{T}$ or $\mathcal{M}$. Figure 7.18 shows the runtimes for computing $x_{1,2}^{(20,20)}$ using the ACR resampling approach (keeping $n_{ACR} = 100k$ constant) and the reverse approach, respectively. For both cases, the ACR construction times are not included, since at query time only the precomputed ACR histograms need to be accessed and processed. For the following experiments, ACRs with $\alpha = 10, 20, 40,$ and 100 bins are applied. For resampling, the runtimes are constant at about $120ms$ for $10, 20,$ and 40 bins, due to the constant number of inversion steps ($2 * n_{ACR}$). The increasing runtimes for $\alpha = 100$ are mainly due to the higher loading costs, which dominate the runtimes as the size of the ACR increases. For the reverse processing approach, the costs are lower than for the resampling approach for all cases. The runtimes are slightly increasing with the number of α. This is due to the increase of required computations for each additional ACR bin (a similar increase occurs with larger β).

For comparison, the computation times required by the pure sample-based approach has also been evaluated, using $5k, 10k$ or $40k$ samples. For the sample-based approach, the times for copula construction and application were summed up. As a reference, the resulting execution times for the sample-based approach are shown next to the ACR-based results, even though it must be noted that the numbers of α and n_S are not strictly comparable. Therefore, the graph rather associates comparable degrees of result accuracy for the two approaches. For example, the results of applying an ACR with $\alpha = 20$ is compared to the application of a copula with $n_S = 10k$. One can observe that the reverse ACR-based computation runs three times faster than the sample-based approach for comparable degrees of accuracy at $n_S = 10k$ (for comparison, see the accuracy results for $\alpha = 20$ and $n_S = 10k$ in Figure 7.14). The runtimes for the resampling-based ACR approach are higher than those for the sample-based approach up to $\alpha = 20, n_S = 10k$, but then stay at an almost constant low level.

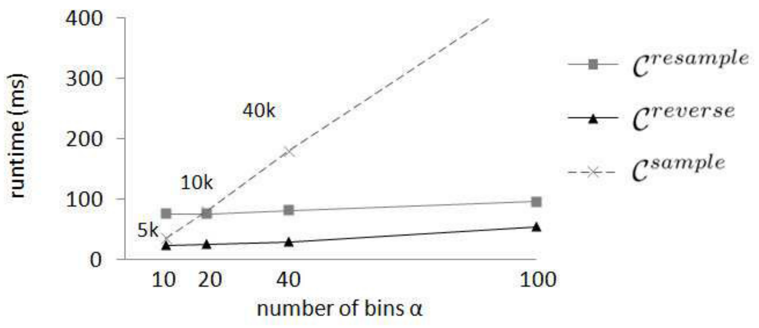


Figure 7.18: Runtimes (in ms) for deriving $x_{1,2}^{(20,20)}$

Overall, the sub-second runtime results indicate that the ACR-based approach is applicable for ad-hoc queries even when high result accuracies are required.

Although this thesis addresses mostly the handling of bi-dimensional correlation structures for ad-hoc analyses, the issue of higher-dimensional correlation patterns is relevant, and is therefore addressed in a further experiment. Higher-dimensional ACRs (i.e., m-dimensional histograms where $m > 2$) incur larger costs for loading as well as for correlation processing. One benefit of the ACR approach is the possibility to process subregions (i.e., sub-cubes in the multi-dimensional case) of the ACR independently. Figure 7.19 shows the processing costs for the reverse ACR processing scheme when correlating $m = 3$ and $m = 4$ marginals using an m-dimensional ACR. In the figure, the results for the single-threaded execution are shown next to executions using t threads, each loading and processing a t^{th} subregion of the ACR. Due to the costs for initial loading and setup as well as the final merging of results histograms, the processing costs do not scale linearly with the number of parallel threads. Still, using parallelization one can achieve ad-hoc results (well below $1s$) even in the case of higher-dimensional correlation structures.

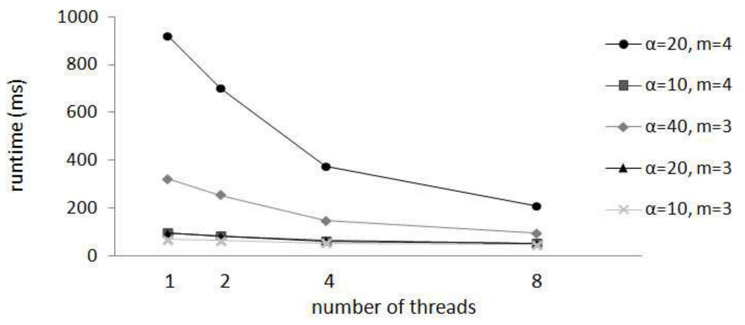


Figure 7.19: Parallel execution of $\mathcal{C}$ (reverse) for correlating $m = 3$ and $m = 4$ distributions

Note that, different from the ACR approach, the runtimes for the native sample-based approach grow only linearly in m for *constant* numbers of n_S. However, as the dimensionality of the sample space grows, it is necessary to increase n_S exponentially in m in order to achieve similar sampling accuracies . This cost increase affects also the ACR resampling approach, which results in processing times of more than $7s$ when using a 3-dimensional ACR. As the number of bins α is usually much smaller than the number of sample points per dimension, the costs for the reverse processing scheme increase less than for the resampling approach.

Optimization As described in Section 4.5.1, one can further optimize query processing by exploiting the specific characteristics of ACRs. To this end, it is necessary to have available information about a complex query (involving a sequential application of $\mathcal{C}$ and $\mathcal{M}$ or $\mathcal{T}$) in advance. This is the case when the query is specified completely *before* execution (rather than applying the operators in a step-wise fashion). Further, such information is available when analysis results shall be *recomputed*; this aspect refers to the efficient recomputation approach discussed in Chapter 5, which is evaluated in the next section.

The following experiment evaluates the times required to compute tail probabilities p_{t_1,t_2}, varying both the numbers of bins representing x_1 and x_2 and the applied ACRs (β and α, respectively). As Figure 7.20 shows (using logarithmic scale), optimizing the query execution by rewriting the query such as to compute the tail probability directly over the applied ACR has a major effect on the runtimes. Note that for all cases of the optimized approach, the large computation times induced by high numbers of β (i.e., $\beta \geq 100$) are completely mitigated. This effect occurs because only the applied ACR needs to be processed, rather than first computing the joint distribution histogram $x_{1,2}^{(\beta_1,\beta_2)}$.

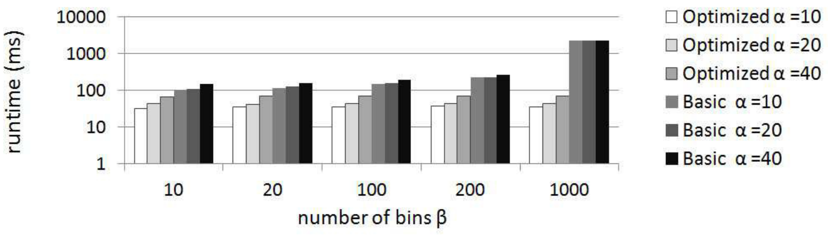


Figure 7.20: Runtimes for computing p_{t_x,t_y}

7.5.4 Summary

In summary, the conducted experiments regarding the handling of correlation and the analysis of multivariate distributions show:

- The obtained query results are highly accurate for different correlation structures and input distribution characteristics.
- Both the presented operations for extraction and introduction of correlation structures perform efficiently, especially when the number of correlated variables or the granularity of the applied ACRs or the marginal distribution histograms is sufficiently low.
- The reverse processing scheme implemented by $\mathcal{C}^{reverse}$ performs particularly efficiently. The calculation of $\mathcal{M}$ and $\mathcal{T}^m$ can be similarly sped up through a query rewriting that exploits a reverse mapping from the multidimensional distribution to the underlying correlation structure.
- For higher dimensional joint distributions and high granularities of the underlying histogram representations, the runtimes can be decreased through parallelization of the $\mathcal{C}$ and $\mathcal{E}$ operators.

7.6 Recomputation

This section discussed the runtime efficiency of the approach for efficient recomputation, which was discussed in Chapter 5. Similar to the setting used previously, experiments are conducted on a server equipped with a Quadcore CPU 2.0GHz and 32GB RAM running SLES

11.0 and queries are evaluated against the TPC-H dataset generated with a scaling factor of $s = 1$. The data was adapted to include uncertain attribute value information. In particular, the variables x_{ship}, x_{rec}, x_{price}, x_{disc} are used, each following the distributions of attributes `l_shipdate`, `l_receiptdate`, `l_extendedprice`, and `l_discount`, respectively (each distribution is internally represented by a histogram with $\beta = 20$) .

First, the performance of recomputing individual operators is evaluated. Both the exact and approximate recomputation of results (using Δ^H and Δ^T) under different modification degrees f_X are considered and discussed for the operators. Further, selected queries from the TPC-H benchmark (namely, queries Q3, Q4, and Q6) have been adapted to using probabilistic instead of deterministic values for selected attributes, such that they become what-if queries over potential future developments. To this end, for each lineitem random variables x_{ship}, x_{rec}, x_{price}, and x_{disc} are associated with the attributes `l_shipdate`, `l_receiptdate`, `l_extendedprice`, and `l_discount`, respectively. The distribution of each such variable is derived by computing a histogram with $\beta = 20$ over the historic attribute values of the respective attribute, grouped per `l_suppkey`. This results in $10k$ distributions, each of which is internally represented and processed through a histogram with $\beta = 20$ bins. For the date attributes, an integer representation is first computed from the values in the TPC-H dataset and a uniform distribution is built for each of x_{ship} and x_{rec} under the constraint that $\bar{x}_{ship}$ (the high support of x_{ship}) is lower than $\underline{x}_{rec}$ (the low support of x_{rec}). The generated probabilistic data are then considered as hypothetic future values of the `l_extendedprice`, `l_discount`, `l_shipdate`, and `l_receiptdate` attributes, and scenarios are computed over this hypothetic future data, as follows:

- **Query S3** computes the potential revenue from orders after *1995-03-01* which are *likely* not shipped until 30 days from the largest order date (now). That is, the probabilistic selection operator is applied over the uncertain ship dates of all selected line items as $\sigma^\tau(X_{ship}, (>, (now + 30)), \tau)$, where the selection probability threshold is set to $\tau = 0.5$. The total potential revenue is then computed by aggregating the price values associated with the line items that passed σ^τ, i.e., $revenue = \mathcal{A}_{SUM}(X_{price})$ for each order. The query is recomputed after varying different fractions of X_{ship}.
- **Query S4** counts the orders containing line items that are *likely* to be received late, i.e., those line items that pass $\sigma^\tau(X_{rec}, (>, l\_commitdate), \tau)$, with a selection probability threshold $\tau = 0.5$. The query is recomputed after varying different fractions of X_{rec}.
- **Query S6** resembles Task A of Use Case 1, which was revisited in Section 5.1. Different from the original TPC-H query Q6, where the goal is to compute potential (past) revenue gains based on the assumption that certain discounts are not granted, we now consider the effect of granted discounts on the potential *future* revenues. The basic assumption is that revenue (computed based on the x_{price} attribute) has a positive correlation with discounts. For each supplier, the variables $x_{price,i}$ and $x_{disc,i}$ reflect the distribution of prices and discounts granted, and a joint distribution $x_{disc,price,i} = \mathcal{C}(C, x_{disc,i}, x_{price,i})$ of the price and discount variables is computed with an assumed correlation structure C (e.g., $C = C_{G,0.7}$). The query then computes the *marginal distributions* $x_{price|\omega_{disc}}$ under a predicate ω_{disc} on the distribution of x_{disc}. Finally,

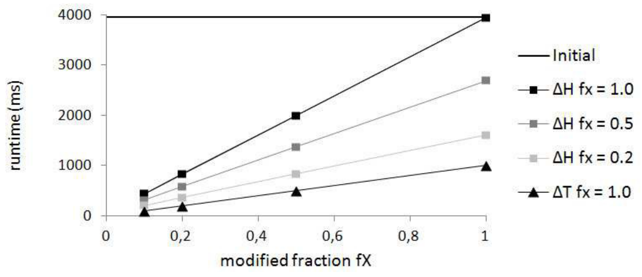


Figure 7.21: $\sigma^{\mathcal{T}}$ using Δ^H/Δ^T for varying f_x,f_X

all marginal distributions are aggregated into a total revenue number, by computing $\mathcal{A}_{SUM}(\mathcal{M}(X_{price|\omega_{disc}}))$. Different fractions of all $x_{disc,i}$ are modified and the query is recomputed.

For each of the operator and query evaluations an initial evaluation is followed by a recomputation based on different degrees of modification f_x and f_X.[3]

7.6.1 Runtimes

First, the efficiency of recomputing individual operators over a set of input distributions X_{price} ($|X_{price}| = 10k$) is evaluated. Then the recomputation of TPC-H queries S3,S4, and S6 is addressed.

The following experiments reflect both the optimization effects achieved by restricting the recomputation effort to impact-relevant inputs as well as the optimization of multivariate analyses based on knowledge about corresponding dim- and cor-components. In both cases, the information represented by the captured $\approx$- and $\hookrightarrow$-relations and intermediate computation results is exploited.

Probabilistic Selection Figure 7.21 shows runtimes for (re)computing $\sigma^{\mathcal{T}}$ over X_{price}. As the costs for testing Φ are similar to computing $\mathcal{T}$ from scratch, the efficiency of recomputing $\sigma^{\mathcal{T}}$ is similar to the initial computation in cases where f_X and f_X are close to 1. In cases of small f_X those x_i without an associated Δ can be skipped in the recomputation process, simply assuming their initially computed results as the results of the recomputation. Further, for small f_x, loading Δ_x^H is cheaper than accessing the complete representation of $x^{(\beta)}$. The savings are not fully proportional with $(1 - f_x)$, due to the overhead associated to initial setup costs for loading and evaluating the delta information. When $f_X \cdot f_x \to 1$, slightly higher runtimes can be observed for the recomputation compared to the initial computation due to the overhead caused by predicate evaluations. Approximate recomputations using Δ^T incur

[3]Without restriction of generality, in our evaluation items x' deviate from their initial version x in $[l_x, l_x + (h_x - l_x) \cdot f_x]$

the lowest costs in all cases due to the small overhead of accessing and processing the Δ^T-parameters.

Aggregation Figure 7.22 shows the runtimes for (re-)computing $\mathcal{A}_{SUM}$ over $X_{extprice}$; again, the modification fractions f_X and f_x are varied. In the first case (denoted by Δ^H in the figure), the set of aggregated items remains the same; only deviations in the underlying distributions are incorporated, resulting in runtimes well below the initial computation in all cases but for $f_X = f_x = 1$. In the second case (denoted by Δ^H+), part of the modification (20% of all $x \in X_\Delta$) result from items added and/or removed to X (e.g., as the result of a previous application of σ^T). Processing newly added items causes higher costs, since both their initial values and associated delta information must be accessed; therefore, in the worst case, the recomputation costs of $\mathcal{A}_{SUM}$ may exceed those of the initial computation. As above, the use of Δ^T is the most efficient alternative due to the low loading costs and the cheap evaluation of deviations at discrete points of Δ^T.

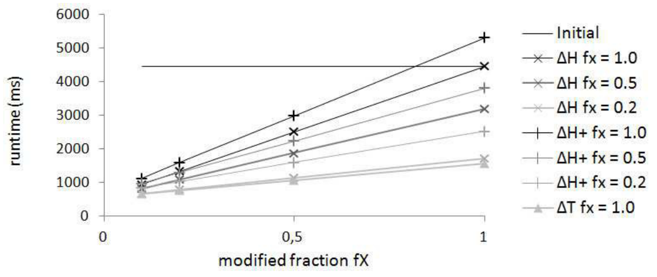


Figure 7.22: $\mathcal{A}_{SUM}$ using Δ^H/Δ^T for varying f_x,f_X

Figure 7.23 illustrates the runtimes for $\mathcal{A}_{MAX}$. One can again see that for small f_x and f_X, the recomputation effort is well below the initial computation times; in case Φ_i is not satisfied for x_i, $x'_{q,i}$ does not need to be recomputed. In case of small f_x this applies for many x'_i, since those do not deviate from x_i at the position of $\dot{x}_{q,i}$. Recall also that a sorted result list of the k highest values $x_{q,i}$ is kept in memory. For all items but those initial k maximum items, $x'_{q,i}$ only needs to be recomputed when $\Delta_{x_i}(x_{max}) < 0$, since only in those cases $x'_{q,i}$ exceeds $x_{q,i}$; this results in a further decrease of required computations. In cases where $f_X \to 1$, one can see that the recomputation is clearly less efficient then the initial computation due to the need to recompute quantile values for almost all x'_i (in addition to the provenance evaluation). Such worst cases can be mitigated by applying the standard computation whenever f_X exceeds a defined threshold, say 0.5.

Multivariate Data Figure 7.24 illustrates the increased efficiency of recomputing $\mathcal{T}^m$ and $\mathcal{M}$, $m = 2$ over $10k$ joint distributions $x_{1,2} = \mathcal{C}(C, x_1, x_2)$ for different fractions f^ϕ. The results reflect the effect of skipping the recomputation of $\mathcal{T}^m$ and $\mathcal{M}$ for those base data where deviations of the involved marginals dim_j do not satisfy Φ_j. Further, even for $f^\phi = 1$, i.e., when all $10k$ results need to be recomputed, the resulting runtimes are well below the initial

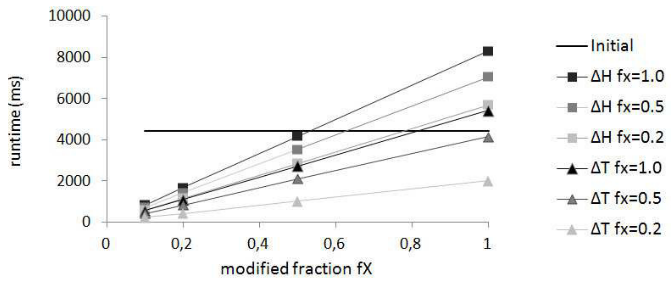


Figure 7.23: $\mathcal{A}_{MAX}$ using Δ^H/Δ^T for varying f_x,f_X

computation due to the described query rewriting approach based on corresponding dim- and cor-components of the multivariate distributions.

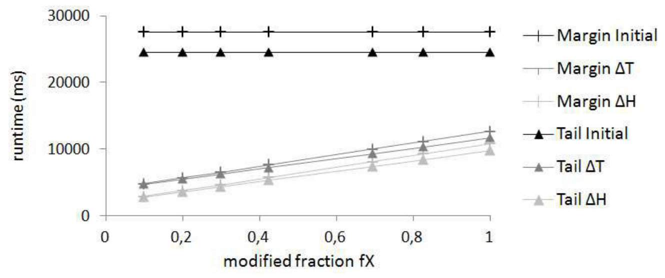


Figure 7.24: $\mathcal{C}$, $\mathcal{T}^m$ using Δ^H/Δ^T for varying f^ϕ

TPC-H What-if Queries

Figures 7.25 and 7.26 display the initial computation times and the runtimes for the re-computation of queries S3, S4, and S6. For queries S3 and S4 (see Figure 7.25) one can observe that the overhead is close to zero for all recomputations, i.e., the incurred runtimes amount to a fraction f of the initial query times. In the case of S6 (displayed in Figure 7.26 along with the other results involving multidimensional analysis), the rewriting approach causes the largest decrease of the observed computation times. Thereby, the resulting query times are well below the initial computation even when f^ϕ (the fraction of all marginals that need to be recomputed due to changes in either of the two dimensions of the analyzed joint distribution) equals 1. Further, for each of S3, S4, and S6, one can again observe that the lowest runtimes are caused by the approximate recomputation (based on Δ^T).

7.6.2 Memory

Storing the provenance graph information on the level of nodes rather than individual (potentially large numbers of) items decreases storage overhead, implying only $|o.in|+|o.out|$ edges

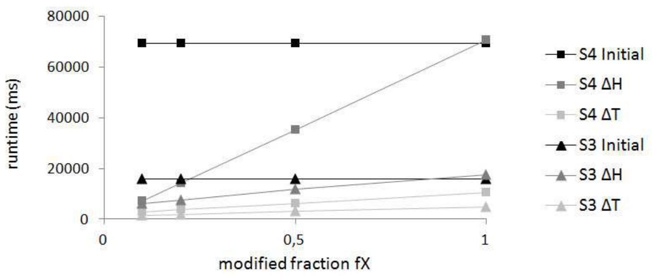


Figure 7.25: Recomputation S3, S4 using Δ^H/Δ^T

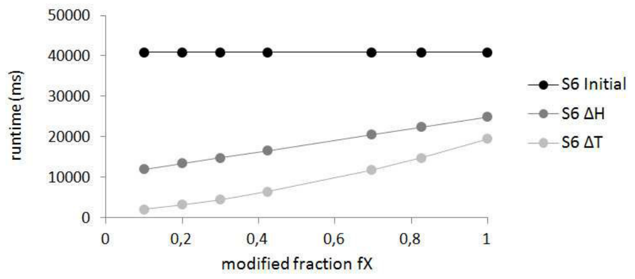


Figure 7.26: Recomputation S6 using Δ^H/Δ^T for varying f^ϕ

$o \overset{used}{\longrightarrow} x \in o.in$ and $d_{out} \overset{wgb}{\longrightarrow} o$. Similarly, $\approx$ and $\hookrightarrow$ edges are stored on the level of in- and output nodes d_{in} and d_{out}.

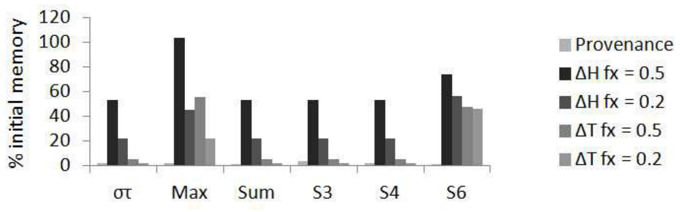


Figure 7.27: Memory consumption for recomputations ($f_X = 1.0$), relative to initial query memory consumption

For an efficient recomputation, it is necessary to keep additional item-level information about initial (intermediate) results in memory (values $\dot{x}_{q,i}$, $kTree$ for $\mathcal{A}_{MIN/MAX}$, and a selection flag and probability $\dot{p}_i$ for σ^τ). Figure 7.27 shows the runtime memory requirements when re-evaluating different operators and queries *S3, S4*, and *S6*, assuming a dataset update fraction of $f_X = 1$ (i.e., all base data is modified). The figure shows the memory requirements M at re-computation time in relation to the memory required for an *initial* computation, (using the same underlying data characteristics). The memory required to keep the recomputation-relevant predicates Φ and intermediate aggregation and selection data ($\dot{p}_i$, $\dot{x}_{q,i}$, $kTree$) are below 4% of M for all operator applications and queries. Furthermore, for $\sigma^\tau, \mathcal{A}_{sum}$, and queries $S3$ and $S4$, the memory costs for delta information Δ^H required in the recomputation process is equal to the fraction f_x of the memory size for initial distribution histograms, as one may expect. In the case of $\mathcal{A}_{MAX/MIN}$, the relative memory requirements are clearly larger compared to an initial computation, due to the fact that the recomputation of quantile values requires that the initial histograms of x_i are loaded and processed *in addition* to the deltas ($\Delta^H_{x_i}$, $\Delta^T_{x_i}$). The figure depicts the case where roughly 50% of all $\dot{x}_{q,i}$ need to be recomputed, resulting in memory costs that exceed the initial costs when $f_x \geq 0.5$. Similarly, for query *S6* we can observe higher relative memory requirements for recomputation. The reason for this is that (most of) the correlation information, represented through x_C, is associated with several joint distributions and thus, must be loaded even when only a fraction of $Y_{disc,price}$ need to be recomputed.

7.6.3 Summary

In summary, the evaluation of the provenance-based re-computation shows:

- The computational runtime costs can be decreased by selectively recomputing complex calculations only for items with impact-relevant deviations, based on an evaluation of Φ_i. This lowers the costs for re-computation in all cases but for $f_X \cdot f_x \rightarrow 1$, and leads to recomputation times roughly proportional with the degree of modification f_X in most cases.

- Specific knowledge about multivariate data, encoded through $\approx$ edges, can be exploited to enable highly efficient multivariate analyses based on the rewriting of the underlying query, when applying $\mathcal{C}$ and $\mathcal{T}^m$ or $\mathcal{M}$.

- The approximate analysis of (assumed) deviations is highly efficient, in all cases but for very high degrees of modification in the case of $\mathcal{A}$. Therefore, using approximate deviation functions Δ^T offers an efficient alternative for evaluating assumed deviations in underlying input data.

7.7 Comparison With MC-based Solutions

As stated in the discussion of related research, MC-based approaches to uncertain data management are closest to the context of this thesis, as they enable a flexible analysis of what-if scenarios. Similar to our work, those approaches address the analyses of arbitrary (multivariate) distributions (with a specific focus on risk analysis in [JXW+10]). This makes them the most suitable candidates for a comparison. The next sections discuss a sampling-based integration of the copula-based correlation introduction with those solutions as well as the implementation of selected aspects of the discussed use cases.

7.7.1 Extending MC-based Systems for Correlation Support

The copula approach to correlation handling, which builds the basis for the ACR approach discussed in Chapter 4, can be integrated naturally with the approaches that rely on MC simulation such as MCDB and PIP. This section briefly describes how those systems can be extended to allow queries that introduce correlation information to data. As a prerequisite, suitable (copula) sampling or generation functions need to be provided at runtime (e.g., through a statistical library such as GNU Scientific Library (GSL)).

Implementing `CorrelateVG` In MCDB, a correlation introduction operation (analog to $\mathcal{C}$) can be implemented through a VG function, denoted as `CorrelateVG`. This function takes provided parameters, which describe the marginal distributions and the desired correlation structure, as input. The implementation of the `CorrelateVG` function then takes care of computing the copula samples and, for each of the samples, the quantile values of the two desired marginal distributions. Using this approach, the introduction of correlation essentially proceeds exactly as discussed in Section 4.2.3. Listing 7.3 shows how the appropriate copula function is first selected to compute a copula sample `(u,v)` based on the input parameters of the desired copula (`H,d`). Then, the parameters of the desired marginals x_1 (`d1,p11,p12`) and x_2 (`d2,p21,p22`) are used to compute a correlated pair of values for each copula sample through inversion. Finally, all samples of the joint distribution are returned. Similarly, one can use a VG function for deriving empirical copulas; we omit further implementation details at this point.

Listing 7.3: Implementation of the CorrelateVG function in MCDB

```
outputVals(d1, d2, p11, p12, p21, p22, H, d)
 float u, v, x, y = 0;
 float[2] vals;
 switch (H){
   case GAUSSIAN:
    x, y =  bivariate_gaussian(1, 1, d);
    u = cdf_gaussian(x, 1);
    v = cdf_gaussian(y, 1);
   case T:
    ...
 }
 switch(d1){
   case GAMMA:
    vals[0] = gamma_inv(u,p11,p12);
   case GAUSS:
    ...
 }
 switch(d2){
   case GAMMA:
    vals[1] = gamma_inv(v,p21,p22);
   case GAUSS:
    ...
}
return vals;
```

In PIP, one can implement a multidimensional sampling function in a similar form, and thus, enable the application of copulas that are sampled at runtime. While MCDB allows to return an array of values, PIP internally represents distributions through the `pip_var` type; with attributes `vid`, which holds the variable id which is shared between correlated variables, and `subscript`, which distinguish between the different variables of the joint distribution. Associating various samples with a common value for the `vid` field ensures that the generation function uses a common seed value for generating the correlated variables, and prevents the optimizer from treating the correlated variables as independent. The generated pseudo-random values for the variables of a joint distribution will therefore be deterministic for a specific pair of variables. The subscript is provided as input parameter in the query, in order to determine which of the two dimensions should be returned. Similarly to the example of the above `CorrelateVG` function, one first must pass the parameters describing the desired type and correlation degree of the correlation structure as well as the marginal distributions. After passing the parameters through a function `pip_CorrelateVG_in`, they are used in the user-defined sampling (generation) function. There, one first draws samples from an appropriate copula function. Second, the inverse cumulative distribution functions of the desired marginals are used to create the samples of the result joint distribution, similar to the `CorrelateVG` function above.

The described custom-implemented VG functions enable a user to define queries that introduce correlation information to sampled data, based on the native copula approach.

7.7.2 Alternative Use Case Implementations

This section briefly describes alternative, MC-based implementations of selected tasks from the use cases of Section 1.2. After a brief discussion of Use Case 1, the focus will be on the implementation of Use Case 2, Task B.2, demonstrating the `CorrelateVG` function described above.

Use Case 1, Task A This task implements a basic sales projection for a new product (or several products), based on historic reference sales data. It can be easily implemented in the MC-based approaches by deriving the parameters of symbolic distributions and computing the expected values of (linear) functions that describe the revenue increase. Therefore, a detailed description is omitted here. The result of the task is a table `Prospect_Sales (pid, value)`, which holds the IDs of the products with their associated monthly sales forecast values.

Use Case 1, Task B This task implements an aggregation of the previously prospected monthly sales values within an aggregation interval, based on information about indeterminate sales periods of the considered product(s).

As input to the query we assume the table `Prospect_Sales (pid, value)` from the previous task, and the table `Releases(pid, ref_id, start, dur)`. Latter holds the IDs of the to-be released products and their reference products, as well as the start and duration variables of the sales period. The implementation of this task cannot be performed directly, due to the lack of a specific operation for interval-based aggregation in the MC-based solutions. Therefore, the computation of interval overlaps is implemented through a corresponding `OverlapVG` function (even though this is not the original intend of the VG concept). This function is then applied to compute the overlap of each of a large number of sampled sales periods with the aggregation interval, $\underline{T}$. That is, for each sampled start and duration time associated with a product release (`r.start, r.dur`) one computes the overlap with the desired aggregation interval (e.g., $\underline{T} = [1, 6]$). Listing 7.4 shows the corresponding query, which first applies the `OverlapVG` function based on the parameter values from `Releases`, and then computes the sum of prospected sales values during the interval $[1, 6]$ based on the computed interval overlaps and the prospected monthly sales computed in Task A of Use Case 1.

Listing 7.4: MCDB example query: Temporal aggregation

```
WITH Interval_Overlaps AS Overlap (
  SELECT r.start, r.dur, 1, 6
  FROM Releases r
)
SELECT SUM (o.overlap * p.value)
FROM Interval_Overlaps o, Prospect_Sales p
```

Use Case 2, Task B.1 To implement this task, the `CorrelateVG` function is applied to define an arbitrary joint distribution in MCDB. For example, the calculation of joint risks in Use Case 2 is implemented by first using `CorrelateVG` to generate samples for $x_{f,h}$ and then computing $p^u_{15\,bn,15\,bn}$ as the fraction of those samples where both x_f and x_h exceed the specified threshold of $\$15\,bn$. Listing 7.5 shows how such a query can be implemented (denoted in SQL style similar to [JXW$^+$08]). For each of $20k$ tuples in a table `claims`, a pair of values (`h_value`, `f_value`) for hurricane- and flood-related claim values is generated using `CorrelateVG`. In the example query, the type of each of the marginal distributions is specified as a `GAMMA` distribution whose (scale and shape) parameters are provided as fixed input parameters. Further, the desired copula type is specified as a `GAUSSIAN` and the correlation degree is set to `d=0.8`. The resulting samples are assigned to the tuples of the `claims_amounts` table, and the required tail probability is computed as the fraction of those values where `h_val > 15bn` $\wedge$ `f_val > 15bn`.

Listing 7.5: MCDB example query: Correlation

```
CREATE TABLE claims_amounts(id, h_val, f_val) AS
 FOR EACH c IN claims
   WITH claims_occur AS CorrelateVG(
        VALUES (GAMMA, GAMMA, 2, 2.6, 0.2, 8, GAUSSIAN, 0.8))
 SELECT h_value, f_value
   FROM claims_occur

SELECT COUNT(*)/(SELECT COUNT(*) FROM claims)
    AS tailprob
  FROM claims_amounts ca
 WHERE ca.h_val > 15bn AND ca.f_val > 15bn
```

On the one hand, this exemplary integration shows the broad applicability of the copula approach, such as through the use of a custom VG function. On the other hand, it also shows the drawbacks that can arise with the sample-based approach.

First, the sample-based implementation depends on the availability of appropriate statistical functions at runtime for creating the desired copula. Such functions may not be available in a specific statistical package (e.g., GSL only provides the bivariate Gaussian distribution) or costly to access at runtime. In contrast, ACRs are precomputed outside the database, thereby enabling the use of functions from potentially different statistical tools. The runtimes of $\mathcal{C}$ then depend solely on the inversion procedure executed at runtime. Second, using the sample-based approach, one cannot use the optimization approaches discussed in Section 4.5 directly, since both approaches rely on the specific features of ACRs. Similar optimizations of the sample-based copula approach require means for parallel generation of copulas, as well as efficient techniques for sampling from the tails of joint distributions, as discussed, e.g., in [JXW$^+$10]. Third, the implementation of `CorrelateVG` requires that each desired correlation structure is implemented as a choice in the `CorrelateVG` function, plus a similar choice of possible marginals. In contrast, an ACR represents correlation generically. One does not need to care about a specific copula at runtime, since they are represented through a histogram, agnostic of the underlying copula function.

Use Case 2, Task B.2 This task, which involves the derivation of an empirical copula prior to the introduction of dependency using this copula, was not implemented for the MC-based solutions. This is because another custom-implemented VG function, plus additional support for the storage of intermediate computation results would have been required for its implementation in MCDB or PIP. Particularly, based on a set of m-dimensional samples, one would need to (i) derive parameters that describe the marginal distribution of each of the dimensions of the sample space, (ii) compute inverses of each input sample value based on its corresponding (previously computed) marginal distribution and (iii) either process the output samples directly or store them by means of a histogram-based distribution representation.

Summary

The comparison of the use case implementations shows that the functionality provided in the thesis can only partially be implemented (emulated) in the MC-based approaches. While the genericity of the MC-based approach is demonstrated e.g., by means of the `CorrelateVG` function, one can also observe that the alternative (sample-based) implementation yields disadvantages such as the need to access particular copula generation or sampling functions at runtime, or the higher runtime requirements due to the effort for sampling copulas (see 7.5.3). Further, the implementation of functionality for the *extraction* of correlation structures from data (and their subsequent introduction to other data) in MCDB or PIP would require additional functionality that goes beyond the basic MC approach. While such an extension is conceivable, it was not evaluated in the thesis.

As stated in the discussion of related work, the efficient scenario recomputation is similar to the optimized scenario computation technique of the JIGSAW system, with regard to the common application scope of the two approaches. However, the two approaches cannot be directly compared due to the fundamentally different uncertain data representation. Hence, the use case implementations were not compared with respect to the recomputation of scenarios.

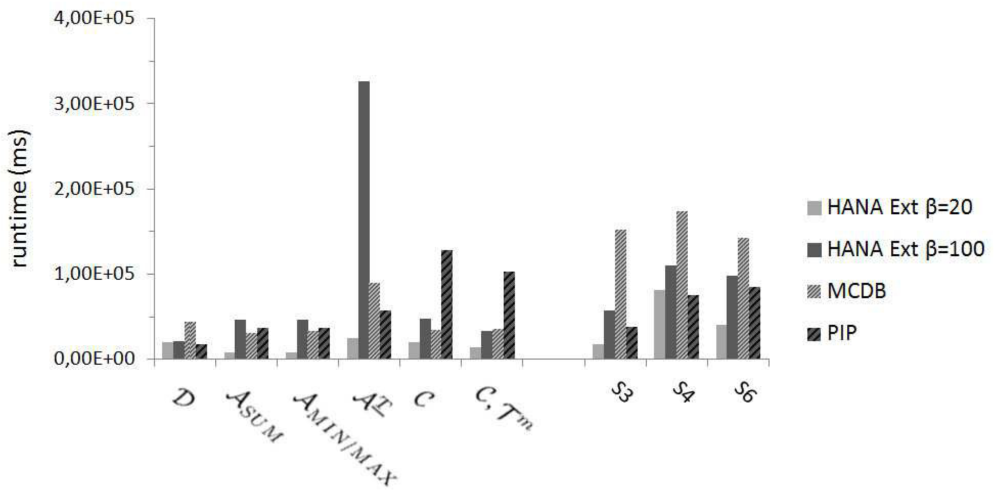


Figure 7.28: Comparison of runtimes for *HANA Ext* and MC approaches

After having discussed the application (and applicability) of MC-based solutions to the approaches discussed in this thesis, the next section compares the corresponding runtime efficiency of the different solutions.

7.7.3 Runtime Performance

This section evaluates the runtimes required by the described prototype system, denoted as *HANA Ext*, and the MCDB and PIP systems, for the execution of individual operators and queries. Similar to the previous evaluation cases, experiments are performed over a dataset generated from the TPC-H benchmark with a scaling factor of $s = 1$. For *HANA Ext*, the distributions associated with each random variable are internally represented through histograms with $\beta = 20$ and $\beta = 100$, respectively. For MCDB and PIP, all sampling processes generate 1000 samples, i.e., each distribution associated with a variable is represented through 1000 samples. The results of the conducted experiments are illustrated in Figure 7.7.3.

The following test cases were performed to compare the runtime requirements of the alternative implementations.

Distribution Derivation As a starting point, the application of $\mathcal{D}$ is evaluated over the set $\dot{X}_{price}$ of all `l_extendedprice` attribute values, grouped by `l_partkey`. That is, $X_{price} = \mathcal{D}(\dot{X}_{price}, hist(\beta), l\_partkey)$ is computed, to derive $N = |X_{price}| = 200k$ distributions. The runtimes are compared against the computation of distribution parameters *mean* and *variance* over the same attribute and grouping in MCDB and PIP. The runtimes achieved by *HANA Ext* for deriving the distributions are at $20s$ for both $\beta = 20$ and $\beta = 100$ (as discussed, the histogram granularity has only a negligible effect on the runtimes of $\mathcal{D}$). The runtimes of PIP are slightly lower at $17.5s$. The runtimes of MCDB are about four times as high, which is most likely being caused by larger times required for loading the `lineitem` table data from disk.

Aggregation The application of the aggregation operators $\mathcal{A}_{SUM}$ and $\mathcal{A}_{MIN/MAX}$ over a data set of $200k$ derived distributions is compared against the results of computing SUM and MAX aggregates for the same set of distributions (using the previously computed *mean* and *variance* parameter values to generate samples for each tuple in the `lineitem` table). The runtimes of *HANA Ext* for aggregating $200k$ variables are clearly the lowest (about $8s$) when a low histogram granularity of $\beta = 20$ is used. The runtimes required by MCDB and PIP (using 1000 samples per generation as noted above) are $42s$ and $37s$, respectively. As can be expected from the previous experimental results from Section 7.4, the runtimes of *HANA Ext* raise (linearly) for a higher histogram resolution of $\beta = 100$, leading to larger runtimes of $46s$ at the benefit of a higher accuracy of the derived histograms.

Temporal Aggregation For a comparative implementation of $\mathcal{A}^{\underline{T}}$ for the MCDB and PIP systems, a function `OverlapVG` is used. The `OverlapVG` function is provided with parameters that define the aggregation interval $\underline{T}$ and distribution parameters for the start and end time distributions. Analogous to the $\mathcal{A}^{\underline{T}}$ operator, `OverlapVG` computes the temporal overlap of the interval $\underline{T}$ with a temporally indeterminate event. i.e., an event with a randomly sampled start and end time. For both the application of $\mathcal{A}^{\underline{T}}$ and `OverlapVG`, the boundaries of $\underline{T}$ are set so that all events lie completely in $\underline{T}$, i.e., $f_\phi = 1$ in all cases. The runtimes required by *HANA Ext* with $\beta = 20$ are clearly the lowest at about $25s$. The runtimes of MCDB and PIP exceed this runtime by a factor of 3 and 2, respectively. With a high granularity of $\beta = 100$, one can observe a clear increase of runtimes for *HANA Ext* (over $300s$) which is due to the quadratic complexity of $\mathcal{A}^{\underline{T}}$, as discussed also in Section 7.4.

Multivariate Analysis The correlation operator $\mathcal{C}^{reverse}$ is evaluated against the described implementation of the `CorrelateVG` function for correlating $10k$ distributions. Correlated samples are drawn for random variables x_{price} and x_{disc} based on the previously computed distribution parameters. A Gaussian copula is used for the correlation, based on a specified correlation degree (e.g., $d = 0.7$ for a copula $C_{G,0.7}$).

For the sequential application of $\mathcal{C}$, one can observe a clear performance benefit for *HANA Ext* compared to MCDB and PIP, with runtimes of about $20s$ and $48s$ compared to $55s$ and $112s$ for MCDB and PIP, respectively. This is mainly due to the efficient reverse processing approach of $\mathcal{C}^{reverse}$ as opposed to the sample-based implementation of `CorrelateVG`.[4] For *HANA Ext* the combined use of $\mathcal{C}$ and $\mathcal{T}^m$ results in even lower runtimes ($14s$ and $34s$, respectively) than the execution of $\mathcal{C}$ alone, due to the optimized processing discussed in Section 4.5. In contrast, the sample-based implementation of *CorrelateVG* for MCDB requires that one first introduces the correlation and then filters the resulting joint distribution samples to meet the conditions of $\mathcal{T}^m$, as described in

[4]The very low performance of the PIP implementation seems to be caused by a workaround for the implementation of `CorrelateVG`, which requires the specification of the correlated variables based on individually parsed parameter strings.

Section 7.7.1. In the case of PIP, the runtimes for the sequential application of $\mathcal{C}$ and $\mathcal{T}^m$ are lower than for the execution of $\mathcal{C}$ alone, since the sampling of the second variable (i.e., x_{disc}) can be avoided in the case that the first variable (x_{price}) satisfies predicate ω_1 of $\mathcal{T}^m$. [5].

Queries S3, S4, and S6 For the three example queries S3, S4, and S6, one can observe that the runtimes of *HANA Ext* for $\beta = 20$ are clearly the lowest for S3 and S6, while they are similar to the runtimes of PIP for S4. In the case of S6, one can particularly observe that the optimized subsequent processing of $\mathcal{C}$ and $\mathcal{M}$ can be exploited, which improves the efficiency of *HANA Ext* compared both to MCDB and PIP. For $\beta = 100$, the required runtimes exceed those of PIP for all three queries due to the larger times required to load and process the fine-granular distribution representations. In $S3$, one can see most clearly the effect of the applied histogram granularity β on the runtimes, due to the computation of the sum aggregate. The runtimes of MCDB are the highest for all cases, which seems to be largely due to the times required for loading the table data from disk.

To sum up the comparison, one can observe that the runtimes are clearly lower for *HANA Ext* than for MCDB in all cases for $\beta = 20$ and even most cases of $\beta = 100$. The runtimes of PIP partially lie below the runtimes of *HANA Ext* (e.g., for the application of $\mathcal{D}$ and in the case of S3 and S4), but clearly exceed the runtimes of *HANA Ext* especially for the case of multivariate analysis using $\mathcal{C}$ and `CorrelateVG`, respectively, which is due to the optimized processing achieved by *HANA Ext* in those cases. A clear exception of the low runtimes of the evaluated functionality is the application of the temporal aggregation operator in case very fine-grained histogram representations are applied, which lead to very high runtimes of *HANA Ext* in the case of fine-granular histogram representations due to the quadratic complexity of $\mathcal{A}^{\underline{T}}$. In the case of the application of $\mathcal{C}$, this negative effect can be largely mitigated due to the use of the discussed optimization techniques, applying the reverse correlation processing approach.

Although the observed runtimes provide an overall insight into the performance of the approaches, it must be clearly noted that an evaluation between the three systems, particularly, between the approaches presented in this thesis and the MC-based solutions, can not yield strictly comparable results. This is because the underlying approaches, the representation of (derived) distributions, and the processing of data differ vastly. An evaluation regarding both the runtimes and accuracy of both approaches relies on different parameters: In the case of the explicit (histogram-based) representation of *HANA Ext*, the parameterization of the histogram granularity is varied, while the performance of the MC-based approaches depends on the number of samples generated per random variable.

Furthermore, the accuracy of results yielded from the execution of the discussed analytical operators is not directly comparable. For example, while allowing both for symbolic and histogram-based distribution representations, this thesis mainly addresses their handling in the form of histogram-based approximations. In contrast, the MC-based approaches rely mostly on the use of symbolic representations of distribution functions, whose parameters are fitted

[5] For MCDB, similarly improved runtimes could be achieved by optimizing the sampling function to draw samples from the tail of the joint distribution. This optimization was however not implemented.

to the data. The fitted distributions can represent the underlying data either with less or more accuracy than a derived histogram, depending on whether the assumed distribution family represents the true nature of the distribution of the data. In general, the accuracy of the base distributions, and therefore the results of any of the considered operators, depends to a large degree on the assumptions about the nature of the underlying data. Therefore, an analysis of accuracies is not included in this evaluation.

7.8 Summary

This chapter presented the evaluation of the concepts presented in this thesis and their implementation by the *HANA Ext* system. First, an implementation of the use cases discussed throughout the thesis was considered. It was shown that the discussed functionality covers the requirements implied by the use cases. The following sections discussed the efficiency of the analytical operators over probabilistic data, including the functionality for the extraction and introduction of correlation in data. The achieved runtimes are largely consistent with the complexity analysis of Chapters 3 and 4. For the operators $\mathcal{C}$ and $\mathcal{E}$, it was shown that their execution can be parallelized effectively in order to achieve runtime improvements. This approach can be applied in order to mitigate the negative effect on runtimes that is caused especially in the case of high dimensionality of the correlation structure (i.e., m-dimensional ACRs with $m > 3$) as well as a high granularity of the applied ACRs. Nonetheless, the focus of the correlation processing functionality discussed within this thesis is on an ad-hoc computation of analyses over a low number of random variables (i.e., $m \leq 3$) with arbitrary distributions. Note again that this is in contrast to related approaches as discussed, e.g., in Section 6.4.2, where the focus is on correlating larger numbers of variables, yet with only a few discrete alternative valuations each. For the bi-dimensional cases, low runtimes of well below one second can be achieved for both the correlation introduction and extraction.

Following the evaluation of the correlation handling functionality, the efficient recomputation of analysis operations was addressed through a number of experiments. Both for the recomputation of individual operators and selected queries that apply those operators, it was shown that the selective recomputation serves to reduce runtimes. In most cases, runtimes are close to proportional to the fraction of modification of the processed data sets. An exception is the application of the quantile-based computation of maxima, due to the higher runtimes induced by processing both the initial distributions and the delta information.

For a comparison with related approaches, the MC-based solutions were chosen due to their similar application context and analysis capabilities. An integration of the correlation introduction operation by means of a `CorrelateVG` function was presented. The discussion and evaluation of this sample-based implementation shows that a flexible handling of correlations can be integrated with MC-based PDBs, yet requires that appropriate statistical library functions are available at runtime to produce different required correlation structures (i.e., different copulas that are sampled or generated). Further, the evaluation of the ACR-based and the sample-based approaches shows that the latter causes higher runtimes. In general, one can observe that *HANA Ext* achieves smaller or comparable runtimes for the evaluation of both individual operators as well as the selected queries. However, it must be noted again that the runtimes of the implementation of *HANA Ext* and the MC-based solutions are not strictly comparable due to the different approaches of the implicit (sample-based) and the explicit representations of uncertainty. This is because the achieved runtimes depend on the number of MC iterations on the one hand and on the granularity of the underlying histogram-based approximation on the other hand.

Considering the selective recomputation based on initially recorded provenance information, no comparison to the MC-based solutions is possible. If a fraction of the symbolically represented input distributions are modified, the complete query must be re-evaluated. That is, for all involved variables one must again draw samples from the associated distributions and evaluate the query for all sampled PWs. In contrast to the basic MC-based solutions, the Jigsaw system (see Section 6.4.5) presents an approach that does enable an efficient computation of scenarios with VG functions under varied parameterizations. This approach provides large performance gains for an efficient exploration of the scenario space, partially reducing the computation times by an order of magnitude. However, again this approach is not directly comparable with the approach applied in this thesis, since the explicit representation of distributions does not allow for a fingerprinting-like technique and vice-versa, the explicit representation of deviations can not be used in the MC-based approach.

8 Summary

To summarize this thesis, this chapter outlines its contributions and lines out the most relevant aspects considering the effectiveness and the efficiency of the presented technologies. A number of interesting further research issues that arose during the time of this work but could not be answered within the thesis will also be addressed as future possibilities of research.

8.1 Conclusion

This thesis addressed concepts for supporting the creation and analysis of scenarios within a scenario planning process at database level. The focus was on the representation and processing of *uncertain* data. First, the context of scenario analysis and planning was introduced in Chapter 1, and the need for appropriate database support for such analyses was motivated. Within this problem space, the aspects of *correlation handling* and the need for *efficient scenario recomputations* were identified as requirements that have hitherto not been addressed sufficiently by existing solutions. Chapter 2 then provided the reader with a conceptual and technical foundation of the application context of scenario planning and the representation and management of uncertainty in data. Based on this foundation, the following section presented the uncertain data model and a set of statistical operations that can be applied as the basic components of a scenario analysis process. Specific focus was then put on the topics of correlation handling (Chapter 4) and the recomputation of analyses based on changed input data (Chapter 5). The presented approaches contribute to and extend the hitherto existing functionality for univariate and multivariate analyses in databases, first, by providing flexible methods for correlation handling and, second, by enabling an optimized processing of the provided analysis functionality for scenario recomputations.

Chapter 6 compared the presented approaches for scenario analysis over uncertain data with prior work. There, we discussed both generic related techniques for uncertainty and provenance management, as well as distinguishing the techniques of this thesis from those used in concrete related solutions for uncertain data management.

Finally, the developed concepts were evaluated in Chapter 7 based on the prototypical implementation *HANA Ext*. An implementation of different use cases demonstrated the applicability of the provided analysis functionality. In particular, the high flexibility of the developed functionality for correlation handling was illustrated. Further, the evaluation of the uni- and multivariate analysis operators proved their runtime efficiency, which renders them suitable for ad-hoc analysis over large amounts of data in most of the considered evaluation cases.

Within the evaluation, specific focus has been put on the presented techniques for correlation handling and scenario recomputation. For the ACR-based introduction of correlation in data, the evaluation showed that the resulting joint distributions are highly accurate compared to the application of the native sample-based copula approach. It was also shown that the resulting runtimes are well-below the sample-based approach, particularly so in the case of the reverse processing scheme, which reverts the inversion approach to map from the bin boundaries of the to-be-computed result histogram to the relevant copula regions. A similar mapping was exploited to optimize the processing of the multivariate operators over correlated data. Further, an effective parallelization was described for both the introduction and the extraction of correlation information based on the concept of ACRs.

The positive effect of selective recomputations over partially modified (deviating) input data could also be demonstrated in the evaluation. Both approaches for the incorporation of histogram-based (known) deviations, as well as approximative (assumed) deviations were evaluated. The results show that both approaches serve to save computation times. In particular, incorporating assumed deviations based on corresponding *approximate delta information* can be achieved in a highly efficient manner.

The focus of this thesis has been on providing support for processing analytical operators over continuously distributed probabilistic data, rather than provisioning a new PDB system. Nonetheless, it was shown that the developed concepts closely relate with and can serve to extend different approaches of existing PDB systems, and especially so to allow a more flexible handling of continuous distributions in probabilistic data. In particular, the ACR-based correlation representation and processing techniques that were presented as a central contribution of this thesis can be integrated with existing PDB solutions. For example, it was shown that one can integrate the processing of copulas or alternatively, the use of generic pre-computed ACRs with the class of MC-based solutions. Similar to the correlation handling functionality, the presented techniques for recomputation of analysis over modified input data apply to continuously distributed data. Thus, it forms a complementary approach to previous work in the field of recomputations and sensitivity analysis over deterministic and discrete probabilistic data.

Concluding, the contributions of the thesis help to extend the applicability of probabilistic data analyses on the database to the continuous case. For many scenario analysis use cases, the provided operators can be applied flexibly as well as efficiently, thus benefiting a flexible, ad-hoc scenario analysis process. Still, there exist a number of open issues as well as restrictions that should be addressed in future to further extend the proposed approaches, and increase their applicability and efficiency.

8.2 Future Research Possibilities

During the discussion and evaluation of the presented concepts several further research questions arose, which could not be addressed sufficiently or were considered out of scope for this thesis. The following aspects are considered interesting fields of future research:

Integrating ACRs and PGMs As stated in the presentation of the ACR approach and its evaluation, the chosen histogram-based representation of ACRs has the drawback that their storage and processing incurs costs that are exponential in the number of correlated variables, m. Although the evaluation showed that analyses of distributions between up to 4 variables incur sub-second runtimes, the exponential complexity restricts the applicability of $\mathcal{C}$ to analyses of joint distributions of up to 5 variables. Comparing ACRs to the related technique of PGMs, one observes that the graphical representation of dependency structures *in principle* is equally complex, in the case that all conditional probabilities between all discrete alternatives of all variables need to be represented. The PGM approach addresses this problem by representing conditionally independent subsets of variables through separate factors thus reducing the overall complexity. This method enables a more efficient representation and processing of dependency structures involving many variables, given that conditional independencies exist among them. A similar approach is conceivable when considering the modeling of dependencies between continuous distributions through copulas, or ACRs, respectively. In particular, considering ACRs as the factors of a PGM, one could similarly target the efficient representation of joint distributions between many variables by exploiting the conditional independencies of subsets, and combining several lower-dimensional copulas (ACRs) in a graph structure. Investigating such an integration of ACRs and PGMs is a promising topic of future research.

ACR Partitioning Schemes The choice of a suitable partitioning scheme for the histogram-based representation of copulas through ACRs has been discussed during Section 4.3. The choice of equi-width histograms in this thesis is based on the fact that their representation and processing incurs the lowest costs compared to more advanced partitioning schemes. A more fine-granular representation of specific regions of the copula, such as their joint tail regions, is achieved by using *nested* histograms. The nested partitioning scheme enables different representation granularities while retaining the efficient storage and processing through statically computed bin boundaries. Further, the use of (nested) equi-width histograms enables a simple parallelization of operations such as the derivation of histograms and the derivation of empirically derived ACR structures, as exemplified in the thesis. In contrast, the use of partitioning schemes that result in arbitrary bin boundaries, such as equi-depth or V-optimal [Ioa03], has not been investigated during the thesis. Important to note, such partitioning schemes could be integrated with most of the presented approaches. Naturally, such an integration would require an adaption of the histogram storage and processing by various operators. Further, the application of different partitioning schemes would hinder the implementation of parallelized and optimized processing schemes, such as for the derivation of empirical ACRs.

Nevertheless, more advanced partitioning schemes can certainly serve to achieve a smaller approximation error over the complete ACR, thus leading to more accurate analysis results, or alternatively enable the copula representation through a smaller total number of bins. Thus, investigating the different partitioning schemes with emphasis on the special characteristics and requirements of the copula approximation is an interesting topic of future work.

Automatic Deviation-based Recomputation An automatic computation of large numbers of scenarios that are “neighbors” of an initial scenario (in terms of similar but partially deviating input data) could be achieved through an automatic scenario recomputation. Such a method can serve as a helpful tool for a user to identify critical (or otherwise interesting) scenarios based on variations in the input data space. Note that this is similar to the parameter space exploration of Jigsaw, yet with a higher emphasis on the nature of individual input distributions rather than parameters of a model. While this thesis addresses an efficient approach for recomputations based on input modifications, an automatic approach for computing a whole space of scenarios was not investigated further. The question whether such an approach can efficiently incorporate and thus exploit the proposed recomputation method is thus subject to future research.

Approximation Schemes for Delta Information The incorporation of assumed input data deviations in the recomputation of scenarios was exemplified in Section 5.3 by means of a triangular approximation function. The evaluation of the proposed approach showed that thus approximated deviations can be incorporated highly efficiently during the recomputation of the analytic operators. Yet, their applicability is restricted to deviations that can be approximated by a triangular function with reasonably low error. Different approximation functions such as more complex piecewise linear approximations could be used, e.g., for a close approximation of actually observed (rather than just assumed) deviations. Evaluating the trade-off between the complexity and accuracy of approximation functions and the resulting runtime efficiency of the recomputation was out of scope for this work. This aspect becomes relevant when considering the recomputation of analyses under a large number of deviations (e.g., in the context of an automatic scenario exploration) for which exact function-based deviations shall be used instead of histogram-based representations.

Evaluation Benchmark for What-if Analysis Finally, during the conceptualization as well as during the evaluation of this thesis, the lack of a suitable set of data which could serve as a realistic benchmark became apparent. The well-known TPC-H benchmark (the generated data set as well as the included queries) is appropriate for analyses over historic data, but does not provide a sufficient basis for evaluating what-if queries as focused in this thesis. Therefore, selected queries of the TPC-H benchmark have been adapted to include aspects suitable for the analyses targeted within this thesis. The design of a dataset and a wholesome set of queries in the application context of planning and what-if analysis would build a better basis both for devising and testing analytic functionalities at the database level, and for evaluating such functionalities from a performance and application point of view. Thus, such a benchmark would certainly constitute a major contribution to the future academic and industrial research in this field.

Bibliography

[ABH09] Daniel J. Abadi, Peter A. Boncz, and Stavros Harizopoulos. Column-Oriented Database Systems. *Proc. VLDB Endow.*, 2(2):1664–1665, 2009.

[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 *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB)*, pages 1151–1154. VLDB Endowment, 2006.

[AJKO08] Lyublena Antova, Thomas Jansen, Christoph Koch, and Dan Olteanu. Fast and Simple Relational Processing of Uncertain Data. In *Proceedings of the 24th International Conference on Data Engineering (ICDE)*, pages 983–992, 2008.

[All83] James F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[AW09] Parag Agrawal and Jennifer Widom. Continuous Uncertainty in Trio. In *Proceedings of the 3rd Workshop on Management of Uncertain Data*, 2009.

[Beu03] Albrecht Beutelspacher. *Lineare Algebra*. Friedr. Vieweg und Sohn Verlag / GWV Fachverlage GmbH, 6th edition, 2003.

[BF05] Rajendra Bose and James Frew. Lineage Retrieval for Scientific Data Processing: A Survey. *ACM Computing Surveys*, 37(1):1–28, 2005.

[BGJ06] Michael H. Böhlen, Johann Gamper, and Christian S. Jensen. Multi-dimensional Aggregation for Temporal Data. In *Proceedings of 10th International Conference on Extending Database Technology (EDBT)*, pages 257–275, 2006.

[BGLS03] Veronica Biazzo, Rosalba Giugno, Thomas Lukasiewicz, and V. S. Subrahmanian. Temporal Probabilistic Object Bases. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):921–939, 2003.

[Bie02] Lauren Bielski. The Great Risk Debate. *American Banking Association Banking Journal*, 94:58–64, 2002.

[BK96] Patrick Bosc and J. Kaeprzyk, editors. *Fuzziness in Database Management Systems*. Physica-Verlag, 1996.

[BKT01] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and Where: A Characterization of Data Provenance. In *Proceedings of the 8th International Conference on Database Theory*, volume 1973/2001, pages 316–330, Berlin, Heidelberg, 2001. Springer.

[Bos06] R. Bose. Understanding Management Data Systems for Enterprise Performance Management. *Industrial Management & Data Systems*, 106(1):43–59, 2006.

[BSHW06] Omar Benjelloun, Anish Das Sarma, Alon Halevy, and Jennifer Widom. ULDBs: Databases with Uncertainty and Lineage. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 953–964, 2006.

[CD97] Surajit Chaudhuri and Umeshwar Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26(1):65–74, March 1997.

[CDLS03] Robert G. Cowell, Philip A. Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems (Information Science and Statistics)*. Springer, New York, May 2003.

[Cok09] Gary Cokins. *Performance Management: Integrating Strategy Execution, Methodologies, Risk, and Analytics*. John Wiley and Sons, 1st edition, 2009.

[CP00] Wes Cowley and Dimitris Plexousakis. Temporal Integrity Constraints with Indeterminacy. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB)*, pages 441–450, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[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.

[DD96] Rina Dechter and R. Dechter. Bucket Elimination: A Unifying Framework for Probabilistic Inference. pages 211–219, 1996.

[DGS09] Amol Deshpande, Lise Getoor, and Prithviraj Sen. Graphical Models for Uncertain Data. In C. Aggarwal, editor, *Managing and Mining Uncertain Data*. Springer, 2009.

[DH05] Xin Dong and Alon Y. Halevy. Malleable Schemas: A Preliminary Report. In *WebDB*, pages 139–144, 2005.

[DS98] Curtis E. Dyreson and Richard Thomas Snodgrass. Supporting Valid-Time Indeterminacy. *ACM Transactions on Database Systems*, 23(1):1–57, 1998.

[DS07] Nilesh Dalvi and Dan Suciu. Efficient Query Evaluation on Probabilistic Databases. *The VLDB Journal*, 16(4):523–544, 2007.

[Eli10] Gal Elidan. Inference-less Density Estimation using Copula Bayesian Networks, 2010. http://event.cwi.nl/uai2010/papers/UAI2010_0054.pdf.

[GD07] Boris Glavic and Klaus R. Dittrich. Data Provenance: A Categorization of Existing Approaches. In *12. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web (BTW)*, pages 227–241. Verlagshaus Mainz, Aachen, March 2007. ISBN 978-3-88579-197-3.

[GG84] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

[GKT07] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance Semirings. In *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems (PODS)*, pages 31–40, New York, NY, USA, 2007. ACM.

[GT06] Todd J. Green and Val Tannen. Models for Incomplete and Probabilistic Information. In *IEEE Data Engeneering Bulletin*, pages 278–296. Springer Berlin / Heidelberg, 2006.

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

[Ham94] Richard Hamlet. Random Testing. In *Encyclopedia of Software Engineering*, pages 970–978. Wiley, 1994.

[Has70] W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and their Applications. *Biometrika*, 57(1):97–109, April 1970.

[IL84] Tomasz Imieliński and Witold Lipski, Jr. Incomplete Information in Relational Databases. *Journal of the ACM*, 31(4):761–791, 1984.

[Ioa03] Yannis Ioannidis. The History of Histograms (abridged). In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, 2003.

[ISW10] Robert Ikeda, Semih Salihoglu, and Jennifer Widom. Provenance-Based Refresh in Data-Oriented Workflows. Technical report, Stanford University, 2010.

[IW10] Robert Ikeda and Jennifer Widom. Panda: A System for Provenance and Data. *IEEE Data Engineering Bulletin*, 33(3):42–49, 2010.

[JLF10] Bernhard Jäcksch, Wolfgang Lehner, and Franz Faerber. A Plan for OLAP. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT)*, pages 681–686, 2010.

[JPK$^+$98] H. V. Jagadish, Viswanath Poosala, Nick Koudas, Ken Sevcik, S. Muthukrishnan, and Torsten Suel. Optimal Histograms with Quality Guarantees. In *Proceedings of the 24th International Conference on Very Large Databases (VLDB)*, pages 275–286, 1998.

[JS99] Christian S. Jensen and Richard T. Snodgrass. Temporal Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):36–44, 1999.

[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 *Proceedings of the 28th ACM SIGMOD/PODS International Conference on Management of Data*, pages 687–700, 2008.

[JXW+10] Ravi Jampani, Fei Xu, Mingxi Wu, Christopher Jermaine, Luis L. Perez, and Peter J. Haas. MCDB-R: Risk Analysis in the Database. In *Proc. of the 36th International Conference on Very Large Data Bases (VLDB)*, volume 3, pages 782–793, 2010.

[KK10] Oliver Kennedy and Christoph Koch. PIP: A Database System for Great and Small Expectations. In *Proceedings of the 26th International Conference on Data Engineering (ICDE)*, pages 157–168, 2010.

[KLD11] Bhargav Kanagal, Jian Li, and Amol Deshpande. Sensitivity Analysis and Explanations for Robust Query Evaluation in Probabilistic Databases. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, pages 841–852, New York, NY, USA, 2011. ACM.

[KN11] Oliver Kennedy and Suman Nath. Jigsaw: Efficient optimization over uncertain enterprise data. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pages 829–840, 2011.

[KY10] I. Kojadinovic and J. Yan. Modeling Multivariate Distributions with Continuous Margins Using the copula R Package. In *Journal of Statistical Sortware*, volume 34, pages 1–20. 2010.

[LAB+06] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific Workflow Management and the Kepler System: Research Articles. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, August 2006.

[LB02] Mats Lindgren and Hans Bandhold. *Scenario Planning: The Link Between Future and Strategy*. Palgrave Macmillan, New York, 2002.

[Leh03] Wolfgang Lehner. *Datenbanktechnologie für Data-Warehouse-Systeme : Konzepte und Methoden*. dpunkt-Verlag, 2003.

[Li06] Jacki Li. Modelling Dependency Between Different Lines of Business with Copulas, 2006. www.economics.unimelb. edu.au/actwww/wps2006/No146.pdf.

[LKA03] G.J.G Lawrie, D.C. Kalff, and H.V. Andersen. Integrating Risk Management with Existing Methods of Strategic Control: Avoiding Duplication Within the Corporate Governance Agenda. In *International Conference on Corporate Governance and Board Leadership*, 2003.

[Ma06] Zongmin Ma. *Fuzzy Database Modeling of Imprecise and Uncertain Engineering Information*. Studies in Fuzziness and Soft Computing. Springer, 2006.

[MFF$^+$07] Luc Moreau, Juliana Freire, Joe Futrelle, Robert E. McGrath, Jim Myers, and Patrick Paulson. The Open Provenance Model. Technical report, University of Southampton, 2007.

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

[Mot96] Amihai Motro. Sources of Uncertainty, Imprecision, and Inconsistency in Information Systems. In *Uncertainty Management in Information Systems*, pages 9–34. 1996.

[MPB10] Paolo Missier, Norman W. Paton, and Khalid Belhajjame. Fine-grained and Efficient Lineage Querying of Collection-based Workflow Provenance. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT)*, pages 299–310, 2010.

[MPS99] S. Muthukrishnan, V. Poosala, and T. Suel. On Rectangular Partitionings in Two Dimensions: Algorithms, Complexity, and Applications. In *Proceedings of the 15th International Conference on Data Engineering (ICDE)*, pages 236–256, 1999.

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

[Nel06] R. B. Nelsen. *An Introduction to Copulas*. Springer Series in Statistics. Springer-Verlag New York, 2006.

[oBS01] The Basel Committee on Banking Supervision. *Consultative Document: Operational Risk*. Bank for International Settlements, Basel, 2001.

[PCA11] Beth Plale, B. Cao, and Mehmet Aktas. Provenance Collection of Unmanaged Workflows with Karma. Technical report, 2011.

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

[Pla09] Hasso Plattner. A Common Database Approach for OLTP and OLAP Using an in-memory Column Database. In *Proceedings of the 35th SIGMOD International Conference on Management of Data*, pages 1–2, New York, NY, USA, 2009. ACM.

[Por85] M.E. Porter. *Competitive Advantage*. Free Press, New York, 1985.

[RS98] Gill Ringland and Peter Schwartz. *Scenario Planning : Managing for the Future*. John Wiley & Sons, 1998.

[SAS10] SAS. Financial Management. Delivering Unified, Reliable Financial Intelligence throughout the enterprise, 2010.

[SBD$^+$09] Michael Stonebraker, Jacek Becla, David J. DeWitt, Kian-Tat Lim, David Maier, Oliver Ratzesberger, and Stanley B. Zdonik. Requirements for Science Data Bases and SciDB. In *4th biennial Conference on Innovative Data Systems Research (CIDR)*. www.crdrdb.org, 2009.

[SBH$^+$09] Anish Das Sarma, Omar Benjelloun, Alon Halevy, Shubha Nabar, and Jennifer Widom. Representing Uncertain Data: Models, Properties, and Algorithms. *The VLDB Journal*, 18(5):989–1019, 2009.

[SD07] Prithviraj Sen and Amol Deshpande. Representing and Querying Correlated Tuples in Probabilistic Databases. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*, pages 596–605, 2007.

[SDG09] Prithviraj Sen, Amol Deshpande, and Lise Getoor. PrDB: Managing and Exploiting Rich Correlations in Probabilistic Databases. *The VLDB Journal*, 18:1065–1090, October 2009.

[SGM93] Richard Thomas Snodgrass, S. Gomez, and E. McKenzie. Aggregates in the Temporal Query Language TQuel. *IEEE Transactions on Knowledge and Data Engineering*, 5(5):826–842, 1993.

[Sir05] C. Sirangelo. *Approximate Query Answering on Multi-dimensional Data*. PhD thesis, University of Calabria, 2005.

[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 *Proceedings of the 28th ACM SIGMOD/PODS International Conference on Management of Data*, pages 1239–1242, 2008.

[SMS$^+$08] Sarvjeet Singh, Chris Mayfield, Rahul Shah, Sunil Prabhakar, Susanne Hambrusch, Jennifer Neville, and Reynold Cheng. Database support for probabilistic attributes and tuples. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering (ICDE)*, pages 1053–1061, Washington, DC, USA, 2008. IEEE Computer Society.

[SPG05] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A Survey of Data Provenance in E-Science. *SIGMOD Record*, 34(3):31–36, 2005.

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

[Wac85] Pierre Wack. Scenarios: Uncharted Waters Ahead. *Harvard Business Review 63*, 5, 1985.

[WMGH08] Daisy Zhe Wang, Eirinaios Michelakis, Minos Garofalakis, and Joseph M. Hellerstein. BayesStore: Managing Large, Uncertain Data Repositories with Probabilistic Graphical Models. *Proc. of the VLDB Endowment*, 1:340–351, 2008.

[WMM10] Michael L. Wick, Andrew McCallum, and Gerome Miklau. Scalable probabilistic databases with factor graphs and mcmc. *PVLDB*, 3(1):794–804, 2010.

[XBE$^+$09] Fei Xu, Kevin Beyer, Vuk Ercegovac, Peter J. Haas, and Eugene J. Shekita. E = MC3: Managing Uncertain Enterprise Data in a Cluster-Computing Environment. In *Proceedings of the 35th SIGMOD International Conference on Management of Data*, pages 441–454, New York, NY, USA, 2009. ACM.

[YG98] Adnan Yazici and Roy George. Fuzzy Database Modeling. *Journal of Database Management*, 9(4):36, 1998.

[ZDH$^+$11] Wenchao Zhou, Ling Ding, Andreas Haeberlen, Zachary Ives, and Boon Thau Loo. TAP: Time-aware Provenance for Distributed Systems. In *Proceedings of the 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP'11)*, June 2011.

Publications

[1] Katrin Eisenreich. Towards an Algebraic Foundation for Business Planning. In *Proceedings of Joint EDBT/ICDT Ph.D. Workshop*, pages 161–169, 2009.

[2] Katrin Eisenreich and Philipp Rösch. Handling Uncertainty and Correlation in Decision Support. In *Proceedings of the 4th Workshop on Management of Uncertain Data (MUD) co-located with VLDB 2010*, pages 145–159, 2010.

[3] Katrin Eisenreich, Philipp Rösch, Volker Markl, Gregor Hackenbroich, and Robert Schulze. Handling of Uncertainty and Temporal Indeterminacy for What-if Analysis. In *Proceedings of the 4th Workshop on Enabling Real-Time Business Intelligence (BIRTE) co-located with VLDB 2010*, pages 100–115, 2010.

[4] Jochen Adamek, Katrin Eisenreich, Volker Markl, and Philipp Rösch. Operators for Analyzing and Modifying Probabilistic Data – A Question of Efficiency. In *Proceedings of BTW 2011*, pages 454–473, 2011.

[5] Katrin Eisenreich, Jochen Adamek, Philipp Rösch, Gregor Hackenbroich, and Volker Markl. Correlation Support for Risk Evaluation in Databases. In *Proceedings of the 28th International Conference on Data Engineering (ICDE) 2012*, Washington DC, 2012.