A UNIFIED FRAMEWORK FOR EVALUATING ALGEBRAIC QUERIES OVER ANNOTATED RELATIONS

Zahra Asadi

A thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

September 2015

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Zahra Asadi**

Entitled: **A Unified Framework for Evaluating Algebraic Queries Over Annotated Relations**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Chair

Dr. Greg Butler

_____ Examiner

Dr. Gosta Grahne

_____ Examiner

Dr. Vangalur Alagar

_____ Supervisor

Dr. Nematollaah Shiri V.

Approved _____
Chair of Department or Graduate Program Director

_____ 20 _____ _____

Dean
Faculty of Engineering and Computer Science

Abstract

A Unified Framework for Evaluating Algebraic Queries Over Annotated Relations

Zahra Asadi

We study concepts and techniques for modeling and processing uncertain relations. Intuitively a piece of data is uncertain if its truth is not established definitely. Similarly, a relation is uncertain when its true state is not ascertained. A major source of difficulties is the semantics of uncertain relations defined based on the notion of possible worlds, which is a set of standard relations one of which represents the true state of the real world data but we don't know which one. This has posed serious challenges for over two decades in database and AI research, however, the topic has gained revived attention in database community again due to some emerging applications such as sensor networks, surveys and imputation techniques, and privacy-preserving data mining applications that require storing and processing such data effectively. Our work is motivated by and concerned with practical issues affected by the exponential number of the possible worlds. We study existing models and techniques and consider the semiring model, a representation model of annotated relations, to represent uncertain relations in our work. Our choice of model is justified for being equipped with an algebra for evaluating queries over annotated relations. We illustrate how the model lends itself to a framework to study models and algorithms for both uncertain relations and probabilistic relations in a unified manner. The ideas and solutions provide a basis for development of a system for annotated data management.

Acknowledgments

First and foremost, I offer my sincerest gratitude to my supervisor, Dr. Nematollaah Shiri for his advice, encouragement, motivation and immense knowledge. Without his guidance and support, this work might not have been completed. Thank you for providing me with an excellent atmosphere and bringing out the best in me. I could not have imagined having a better advisor and mentor for my masters thesis. I am very grateful to my family for everything. I am indebted to my parents for all their scarifies for me since the beginning. It was their love and motivation which helped me in moving forward in life. Finally, I would like to thank my friends Jessica, Mahsa, Raheel, and Soyoung in database research lab for their consistent help and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Uncertain data management has been a focus in database and artificial intelligence research for more than two decades. The field has gained a revived attention due to the emerging of some applications such as data cleaning, data extraction, and sensor application which generate such data [14].

Intuitively a piece of data is uncertain when its truth is not established definitely. Similarly a relation is uncertain when its true state is not ascertained. Compared to standard relations, where any instance of a relation is its true state, an instance of uncertain relation represents a set of standard relations one of which is its true state but we do not know which one. Any member of this set is called a possible world or a possible instance. That is, the notion of possible worlds defines the correct semantics of uncertain relations.

Uncertainty associated with data can be categorized based on uncertainty constructs into attribute-level or tuple-level. In the former, the value of an attribute is

chosen from a set of given values, resulting in different possible worlds representing that uncertain relation. The latter imposes some constraints on the arrangement of the tuples. In the simplest case, the constraint indicates presence or absence of the tuples, and in the more complex cases, is about interdependency amongst the tuples.

The major challenges in modeling and processing uncertain relations stem from their semantics as the number of possible worlds can grow exponentially in the number of tuples. To address the challenges, several representation models and query processing techniques have been proposed.

The proposed models can be categorized into the following two main groups, based on their expressiveness: complete and incomplete models. Complete models are complex and nonintuitive, they can represent any set of possible worlds and they are closed under all relational algebra (RA) operations. Incomplete models on the other hand are simpler and less complex, and they are closed under a subset of RA operations. Based on different representation models, different query evaluation techniques are proposed. Some of them establish the theoretical basis in this field and are of theoretical interest while some are of practical interest [6, 4, 14, 8, 3].

Conventional Database Management Systems (DBMS) are inadequate to uncertain data. A few research prototype systems have been developed to manage uncertain data. Examples of such systems include the Trio system, developed at Stanford University [3], the Orion system at Purdue [16], and MayBMS [7] at Cornell University. Trio considers accuracy, lineage and data as the first class citizen [3], whereas Orion [16] has built-in support for probabilistic data. Both Trio and Orion support

possible worlds semantics. MayBMS [7], on the other hand, is a system for managing large uncertain and probabilistic databases. Despite the progress made in managing uncertain data, building powerful and full fledged systems to manage such data it is still an active research topic.

Intuitively, uncertain relations are defined based on possible worlds semantics. In our work among different definitions accepted for probabilistic relations, we pick the one which defines a probabilistic relation based on possible worlds semantics [4]. This identical semantics helps us to propose a generic framework to evaluate queries over such relations. To realize it, we choose the "semiring model" [6] as an underlying representation model of the framework. However, in order to evaluate queries over probabilistic relations based on our desired semantics, possible worlds semantics, the model has been revised. In fact, the revised semiring model defines the theoretical basis of the framework.This model allows the development of a unified framework to evaluate algebraic queries over uncertain relations in general, and probabilistic relations in particular. We build a running prototype of the proposed framework on top of the PostgreSQL as a conventional DBMS.

1.1 Challenges of Uncertain Data Management

There are a number of challenges in dealing with uncertain data, described as follows.

1. Modeling: the key point is how to capture uncertainty associated with the data while keeping the data as simple as possible. This is to be done in a way that

uncertain data can be managed effectively [1]. In this case, the trade off between intuitive and incomplete models and complete and complex plays a role.

2. Query processing: processing queries over uncertain data or over integration of such data sources poses major challenges. Most of the existing solutions adapted standard query evaluation techniques to uncertain data. The extent to which they could be adapted and the limitations of the existing techniques make this area of research quite interesting and relevant.

3. Uncertain data mining: existing data mining techniques do not have enough support for uncertain data. New algorithms and techniques are needed to take into the account the presence of uncertainty in the data.

Among major challenges described above, our work focuses on two aspects: modeling and query evaluation techniques. These two aspects are related in that the algorithm and solution approaches proposed for processing queries over uncertain data depends on the underlying representation model of the data. We will elaborate more on these aspects and report on prototype development in Chapter 2 on background and related works.

1.2 Approaches to Uncertain Data Management

There are two approaches to manage uncertain data. In one approach, uncertainty is treated as a first class citizen and the DBMS is aware of, and exploits the uncertainty in the data in storage structure and query processing techniques as well. That is,

the query processor is aware of and manipulates uncertainty modeled in the process of query evaluation. This is referred to as a "heavy way" approach. The second approach, called "light way", treats uncertainty as a second class citizen. In this approach a standard DBMS is extended to manage uncertain data. In fact, there is a conventional DBMS as a first layer and then, based on the model used to represent uncertain data, a new layer is built and mounted on top of the first layer. In the following, we compare these two approaches in terms of extendibility and usability features.

1. **Extendibility**: In a "heavy weight" approach to building a DBMS for uncertain data, the system is built from scratch. This means, depending on a desired model picked for the representation of uncertain data a DBMS is developed to support it. While this fresh start provides more opportunities for optimization of representation structures as well as query evaluation, it requires more time, effort, and cost to develop. An important point to keep in mind is that representation models and formalisms of uncertain data may go through frequent changes in order to meet different application needs. Consequently, following this approach to develop and maintain a DBMS to manage new or revised uncertain data models could be very expensive and time consuming in general. In contrast, a "light weight" approach uses an existing conventional DBMS as the first layer and then finds a suitable way to extend and adapt it to manage uncertain data. All the missing and required functionalities will be implemented in the application/second layer built on top of a DBMS. Compared to the first

approach, the light weight approach provides limited opportunities for optimization. The advantage, however, is that the changes in the representation model to support different application needs could be handled with less effort.

2. **Usability** : Database users are familiar with at least one of the conventional DBMSs. Beside, the users may need to deal with both certain and uncertain data. It is more convenient to provide users a framework that seems familiar for them to interact with. Such an opportunity is available with the "light weight" approach. In contrast, DBMS built following the "heavy weight" approach does not appreciate much the existing DBMS and tries to fulfill the tasks by its own. One may argue that uncertain DBMS can subsume standard ones as a special case to encourage existing DBMS users to migrate to the new DBMS. However, it is unclear if such "dual" systems can perform on regular, certain data with desired efficiency.

As part of this research, we follow a "light weight" approach and develop a running prototype of a framework which helps to evaluate queries over annotated data. The framework developed interacts with PostgreSQL DBMS. The architecture of the proposed framework, its components, and the way it interacts with PostgreSQL are presented in Chapter 4.

1.3 Thesis Contributions

1. We developed a generic framework for parsing and generating plans for evaluating queries over annotated relations. This is obtained by extending the relational data model to represent and process annotated data based on the "semiring provenance" formalism [6].

2. We built a running prototype of our proposed framework, which is able to evaluate algebraic queries over different categories of annotated relations, including uncertain and probabilistic relations. We were concerned with correctness, efficiency, ease of implementation and maintenance of the proposed framework.

3. In order to evaluate queries over probabilistic relations, without worrying about the independence assumption, we revised the algorithms proposed in semiring model [6] based on the work of Dalvi *e*t al. [4].

1.4 Thesis Outline

The rest of this thesis report is organized as follows. In Chapter 2, we provide a background and review of related works on uncertain data. This includes a comprehensive study of existing representation models and a review of query evaluation methods. In Chapter 3, we discuss in detail the models defining the basis of our framework, and illustrate their advantages. In Chapter 4, we present the architecture of the proposed framework and its modules, together with technical details of our implementation. In

Chapter 5, we illustrate the query processing steps, and the results produced by the framework implemented. Chapter 6 includes concluding remarks and future plans.

Chapter 2

Background and Related Work

In this chapter we provide a background and review related literature on representation models of uncertain relations and processing queries over such data. We are interested to better understand the balance between the expressive power of models and query processing efficiency. As we will see, an application's needs should be considered to determine this balance.

2.1 Uncertain Relations

We begin with the definition of uncertain relations. A relation is uncertain if its truth is not fully ascertained. An uncertain relation can be encoded by a set of conventional (certain) relations, called possible worlds (instances), rather than by just a single certain relation [13]. While possible worlds is the semantics basis for uncertain relations it is not practical due to the exponentioal number of the possible

worlds with respect to the number of tuples.

To illustrate the problems and issues, consider an instance of an uncertain relation R (A, B, C) in Table 1. Suppose that the existence of some tuples in R is uncertain. Then the standard relational data model without modification cannot represent the simplest form of uncertainy in which presence of the tuples is uncertain. To avoid enumeration of all possible worlds, few interesting and powerful models have been proposed.

For this example our representation model is the extension of the relational data model in which "?" is used to represent uncertainty about the existence of the tuples. This is called Maybe relations [11]. In Table 1, R is a maybe relation which can be considered as an uncertain relation also. Note that column Identifier added to the schema of relation R is used for ease of reference and it is not part of the model.

Table 1: An Instance of Uncertain Relation R(A,B,C)

Identifier	A	B	C	Annotation
t_1	a	b	c	?
t_2	f	g	e	
t_3	d	b	e	?

The set of possible worlds of R is:

$p_1 = \{t_2\}$, $p_2 = \{t_2, t_1\}$, $p_3 = \{t_2, t_3\}$, and $p_4 = \{t_2, t_3, t_1\}$.

The valid question might be: what are the problems of modeling uncertain relations? We used a very simple model and we were able to represent uncertain data. This question can be answered from two different angles: first, uncertainty associated

with data is not always as simple as the existence of the tuples so we need expressive enough models to capture uncertainty of different kinds. Second, we pose queries over uncertain relations. The ideal is that, independent of a user query, the result can be representable by the same model as the input relations. To see how easily "maybe relations" fail let us consider the following query over R, taken from [6].

$$\text{q(R)} = \Pi_{AC}((\Pi_{AB}(R) \bowtie \Pi_{BC}(R)) \cup (\Pi_{AC}(R) \bowtie \Pi_{BC}(R)))$$

The query consists of several subqueries that should be executed in some proper order to yield the correct result. Here we look at the result of the execution of a subquery $\Pi_{AB}(R) \bowtie \Pi_{BC}(R)$, illustrated in Table 2, to highlight the issues regarding to query evaluation of uncertain relations. As before, column "Identifier" is added to the result to facilitate referring to the tuples.

Table 2: The result of $\Pi_{AB}(R) \bowtie \Pi_{BC}(R)$

Identifier	A	B	C	Annotation
t_1	a	b	c	?
t_2	d	b	e	?
t_3	f	g	e	
t_4	a	b	e	?
t_5	d	b	c	?

The uncertainty captured by this model cannot go beyond the existence of the tuples. While we need a model to capture any possible interdependency introduced in the process of query evaluation. We can name an example of such interdependency among the tuples of Table 2: when t_1 and t_2 are present then both t_4 and t_5 should be also present. However, such interdependencies cannot be represented by maybe relation. What we see here is that the model is not closed under RA operations. In

the following section, we discuss desired features of representation models including closure under RA operations.

2.2 Properties of Representation Models

As illustrated through the above example, one of the desired properties of a model is being expressive enough to capture the uncertainty modeled. A natural question related to the expressiveness of the model is that whether every possible set of relation instances can be represented by the model [14]. The positive answer implies ***completeness*** of the model. Whereas, it is said that a model M is ***closed under an operation Op*** when applying Op on any uncertain relation in M produces an output that is representable in M [14]. It can be inferred that if a model is closed under an operator, the reasonable implementation would perform the operation on uncertain relation directly rather than a set of possible worlds, see Figure 1. It shows that if R is an uncertain relation in a model M, I(R) is a set of possible instances, and Op is an unary operator under which M is closed, then there is an implementation so that Op can be applied directly on R rather them every instance of R; dotted arrow says that Op can be applied on R directly following the implementation. For every incomplete model, there is a subset of RA operations under which the model is closed. It also implies that if that subset is sufficient for some applications, it is better to choose that incomplete model for the application at hand rather than a complete model. A complete model, however, is closed under all RA operations because every operation

generates a finite set of instances and any set is representable in a complete model.

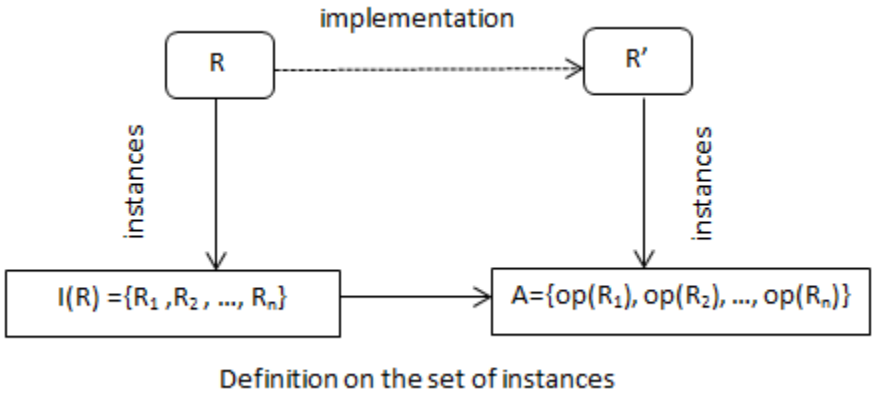

Figure 1: Closure Property [14]

2.2.1 Uncertainty Constructs

Uncertainty associated with the data varies from simple, alternative values for attributes, to more complex that applies some constraints on the arrangement of the tuples [14]. Technically speaking, they are called uncertainty constructs and grouped into two categories: attribute-level and tuple-level [13, 3, 15].

The attribute level construct indicates that the value of one or more attribute(s) is taken from a finite set of values rather than a single value [13]. The following scenario illustrates uncertainty construct at the attribute level [3]:

Scenario1: Some people who witnessed a car accident are talking about their observations, represented in Table 3. However, they are not certain about the model and the color of the observed car. In Table 4 all the possible worlds are enumerated.

The tuple-level uncertainty constructs or existence constraints across tuples, are

Table 3: Witness [3]

Name	Model	Color
bob	toyota	{red,blue}
john	{honda,toyota}	white

Table 4: Possible Worlds of Witness

Name	Model	Color
bob	toyota	red
john	honda	white

Name	Model	Color
bob	toyota	blue
john	honda	white

Name	Model	Color
bob	toyota	red
john	toyota	white

Name	Model	Color
bob	toyota	blue
john	toyota	white

about absence/presence status of the entire tuple or apply some constraints to correlate tuples [1, 13]. The former case, representing the absence or presence of the tuples, gives rise to the so called "maybe" tuples or relations [3, 11], recording lack of knowledge about the true status of the real world fact that the tuple represents for. The latter, by adding some constraints on the tuple identifiers or variables represents the correlation among tuples. The following scenario illustrates uncertainty construct at the tuple level:

Scenario 2: The location of several objects reported by some sensors is represented in Table 5 [2]. To capture such uncertainty, tuples are annotated. The annotations apply some constraints on the tuple identifiers to express the correlation among tuples. As an example, in Table 5, tuples t_1 and t_2 are reporting the location of the same object. The object, a, cannot be located at different locations at the same time so the annotation associated with t_1 forces such constraint.

Possible worlds of the relation Observation are: $p_1 = \{t_2, t_1\}$ and $p_2 = \{t_2, t_3\}$.

Table 5: Obeservation [2]

Identifier	Object Name	Location	Observer Name	Annotation
t_1	a	(x,y)	p_1	$t_1 \oplus t_3$
t_2	b	(x,y+2)	p_1	t_2
t_3	a	(z,y)	p_2	t_3

A desired representation model should help capture uncertainty effectively, for optimized storage structure and query processing. It should also be flexible and adaptable to different applications. In the next section, we study different representation models, in particular focusing on expressive power, closure and completeness properties.

2.3 Representation Models of Uncertain Relations

There have been numerous models proposed for storage structure of, and query processing methods over uncertain data. While the main components in the existing approaches are the same, they differ in detail. The main difference is rooted in the way in which uncertainty is represented and stored which has direct influence on the completeness and closure properties of the model. In what follows, we review major proposed models and contributions.

2.3.1 Conditional Tables

Conditional tables (C-tables) [8] are one of the earliest models proposed to manage incomplete data. In this model, every tuple is annotated with a condition in the form of a logical expression. Since there is no restriction on the logical expressions

and the allowed operators in the annotations, to capture uncertainty, every possible set of relation instances can be represented in this model implying the completeness property of the model. The direct consequence of being a complete model is being closed under all RA operations.

The tuples of the base relations are normally annotated with a simple proposition, a variable. While, in order to capture lineage and correlations, the annotations of tuples in a query result could become arbitrary large and complex being referred as a drawback of the model [3]. Consequently, reasoning about the data gets more complicated, leaving it not suitable in practice.

The idea of using constraints to represent uncertainty proposed in C-tables [8] is the one used in almost all proposed models. However, the research in this field has also concentrated to overcome the drawback of C-tables, being not so practical. This induced a hierarchy of working models, discussed next.

2.3.2 Hierarchy of Working Models

There is an inherent tension in modeling uncertain data: complete models are complex and non-intuitive and incomplete models are more intuitive and simpler [13]. Incomplete models are closed under a " subset" of RA operations. In addition, they can represent "certain" types of uncertainty. So, they might be sufficient for some specific applications. This is the main idea behind the hierarchy of the working models, depicted in Figure 2, [13, 14]. All the nodes, except the top, are incomplete representation models of uncertain data. Based on the various types of constraints

allowed in the annotation, they can capture different types of uncertainty.

In Figure 2, M stands for the model and subscript is the type of constraint allowed in the annotation to capture tuple-level uncertainty. Whereas, the only superscript, A, allows attribute-level uncertainty.

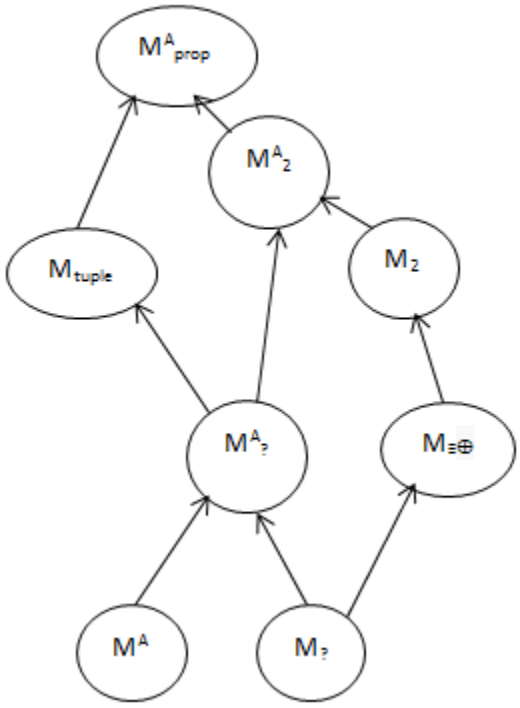

Figure 2: Expressiveness Property of Working Models[14]

At the lowest level, $M_?$ and M^A are the models that capture the simplest type of uncertainty; the former captures the simplest type of uncertainty at tuple-level which is the presence/absence of the tuples and the latter models the only possible type of attribute-level uncertainty. As we go up through the hierarchy, the expressiveness of the models increase; in other words, they can capture more complex type of uncertainty. Finally, at the top level, M^A_{prop} is the only complete model in this hierarchy in

which any logical expression is allowed as an annotation and can model any possible set of relations instances.

An arrow from a node X to a node Y in Figure 2 indicates that the expressive power of these two nodes are comparable. More precisely, Y is more expressive than X. The working models impose certain properties on the possible worlds of uncertain relations, represented by these models, including constant cardinality, path connectedness, unique minimum, etc. [13, 14].

The hierarchy of working models allows the users to trade off between the expressive power and the complexity of the models, in order to pick a suitable model for the application at hand. It also proposes to develop a DBMS consisting of two layers [14]. As the underlying logical layer, there is a complete model on top of which one or more incomplete models are mounted as the working models. An application's needs dictate which working model is more suitable. The working layer provides an abstraction that makes it easier for the users to understand, visualize, and formulate queries [14].

Motivation for Different Working Models

As the tuples in any model M in Figure 2, except for M^A are annotated with a constraint of the form of a logical expression, M represents a kind of C-table. However, the constraints may range from the simplest case of Boolean variables to arbitrary logical expressions. This variation of constraints motivates and justifies the proposed

hierarchy of the working models [14]. Among all models, the one at top has the expressive power identical to C-table as this is the only complete model in the hierarchy. The following example illustrates why we need different working models. In Figure 2, model $M_?$ allows "?" as an annotation. To understand the expressiveness of this model consider the set of possible worlds given in Table 6.

Table 6: Possible Worlds of Registration

Name	Course
bob	comp6521
bob	comp6641

Name	Course
bob	comp6521

Name	Course
bob	comp6641

Recall that $M_?$ captures presence/absence of each tuple only; it cannot capture correlation among tuples. The question is whether there is any relation in $M_?$ whose possible worlds are the relations given in Table 6? Let us have a closer look at the relation Registration, depicted in Table 7.

Table 7: Registration

Identifier	Name	Course	Annotation
t_1	bob	comp6521	?
t_2	bob	comp6641	?

Here are the enumerated possible worlds:

PW(Registration):{ $p_1 = \{t_1, t_2\}$, $p_2 = \{t_1\}$, $p_3 = \{t_2\}$, $p_4 = \emptyset$ }

The set of PW(Registration) is not identical to the possible worlds of Table 6 implying that $M_?$, as a representation model, failed to represent the uncertainty associated with Table 6. Is there any more expressive model in the hierarchy? Absolutely yes. In the worst case, if none of the incomplete models presented in Figure 2 is able

to capture the uncertainty, we can choose the complete model at top to represent the relation. An uncertain relation whose possible worlds are represented in Table 6 is represented in Table 8.

Table 8: Registration as an Uncertain Relation

Identifier	Name	Course	Annotation
t_1	bob	comp6521	$(t_1 \odot t_2) \vee t_2$
t_2	bob	comp6641	$(t_1 \odot t_2) \vee t_1$

It can be concluded that based on the uncertainty associated with the initial data, we might decide on a working model to be the representation model. However, the data would be queried later on, and it may introduce a new types of uncertainty that also needs to be captured. The state transition diagram, depicted in Figure 3, is an answer provided by the idea of the hierarchy of working models to overcome such a problem.

A dotted arrow in Figure 3, from node X to Y labeled with RA operations indicate that X is not closed under those operations. Model Y is then an immediate model which supports those operators. That is, for each incomplete model M and its associated closed set of RA operations, we can find uncertain database D in M and a query Q over D such that the result of Q cannot be represented in M. Indeed, Figure 3 can be viewed as a guide to pick a working model which is expressive enough and can answer application needs in terms of operators it supports.

The idea behind the hierarchy of models is quite interesting, however, it is more application-oriented. In real life, it is not easy to identify an application's needs. Moreover, we can always expect to have new and more needs than expected.

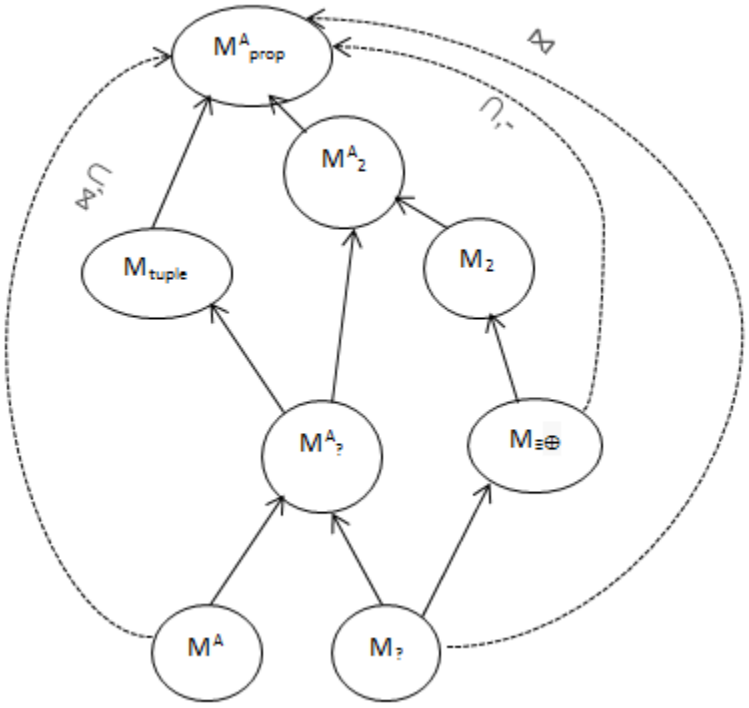

Figure 3: State Transition [14]

Figure 3 has more transition arrows in reality, but we focus more on some basic operations such as join, projection, selection, and union. As an example of drawback of working models is not being closed under join operation. As it's shown in Figure 3, two models at the lowest level are not closed under join and the immediate next model which supports this operation, indicated by using dotted arrows from $M_?$ to M^A_{prop} and from M^A to M^A_{prop}, is the model at top.

We rarely can find any application in which join in not necessary. No support for join operator in any incomplete model, in addition to being application-oriented, makes this model less practical and hence of theoretical interest.

2.3.3 Probabilistic Relations

Here we review another type of relations that can be categorized as uncertain relations. Informally, a probabilistic relation is a relation whose tuples are associated with probability values [4]. Similar to uncertain relations, the precise content of a probabilistic relation is unknown and its standard semantics can be defined based on the notion of possible worlds. We can say that a probabilistic relation is an uncertain relation with probability distribution over the set of possible worlds.

Table 9 shows an instance of a probabilistic relation, in which the annotations are probability values of the tuples.

Table 9: Registration as a Probabilistic Relation

Identifier	Name	Course	Annotation
t_1	bob	comp6521	0.8
t_2	bob	comp6641	0.5

Identical to the uncertain relations, Table 9 encodes a set of possible worlds. Its possible worlds and their associated probability values are listed in Table 10.

Table 10: Possible Worlds of Registration and Their Associated Probabilities

Possible Worlds	Probability Value
$P_1 = \{t_1, t_2\}$	0.4
$P_2 = \{t_1\}$	0.4
$P_3 = \{t_2\}$	0.1
$P_4 = \{\}$	0.1

Note that following possible worlds semantics does not impose any constraint on the probability values of the tuples in the initial table. However, the sum of the probability values of all the possible worlds is always 1.

Our work is motivated by practical issues surrounding modeling and query processing of uncertain relations. One of our goals is to develop a framework to evaluate queries over such relations.

We reviewed existing models to identify a suitable representation model that can capture uncertainty of different types, and it enjoys efficient query processing techniques. We also categorized the probabilistic relations with possible worlds semantics as uncertain relations. However, none of the proposed models can represent uncertain relations as well as probabilistic relations. First, we need to find a way to "relate" the representation models of these two types of relations. This is studied in the next section.

A Representation Model for The Probabilistic Relations

Dalvi *e*t al. [4] propose a representation model for probabilistic relations in which tuples are annotated with event variables rather than probability values. In other words, for a set of possible worlds and their associated probabilities, there is an equivalent probabilistic relation in which tuples are annotated with event variables. The model is a complete model because for any possible set of relation instances and their associated probability values, there is a probabilistic relation. We explain the model briefly in the following.

The model introduces atomic event variables. The number of these variables is one less than the number of possible worlds. There is an assignment algorithm which runs recursively to associate with every possible world a logical expression over the event

variables. It also computes the probability of the event variables. Then, on the basis of membership of tuples to possible worlds, it identifies the logical expression associated with every tuple. Finally, the probability values of the tuples are determined using the probabilities of the event variables [4]. The steps are illustrated in the following example.

The input is the set of possible worlds and their associated probabilities shown in Table 10, and the output is an equivalent probabilistic relation consisting of tuples associated with event variables shown in Table 11. As there are four possible worlds, the number of event variables is three: e_1, e_2, e_3. Then the assignment algorithm is applied which results in, shown graphically in Figure 4:

$fw(P_1) = e_1 \wedge e_2$, $fw(P_2) = \mathrm{e}_1 \wedge \neg e_2$, $fw(P_3) = \mathrm{e}_3 \wedge \neg e_1$, $fw(P_4) = \neg e_3 \wedge \neg e_1$.

While assigning logical expression of event variables to the possible worlds, the probability of event variables is computed.

The probability values associated with event variables are:

$p(e_1) = 0.8, p(e_2) = 0.5, p(e_3) = 0.5$

Finally, based on the membership of tuples to possible worlds, the event variables associated with tuples are generated which results in a probabilistic relation shown in Table 11.

Table 11: Registration Represented in the model proposed in [4]

Identifier	Name	Course	Annotation
t_1	bob	comp6521	e_1
t_2	bob	comp6641	$(\mathrm{e}_1 \wedge e_2) \vee (\neg e_1 \wedge e_3)$

As pointed out earlier, uncertain relations and probabilistic relations are defined

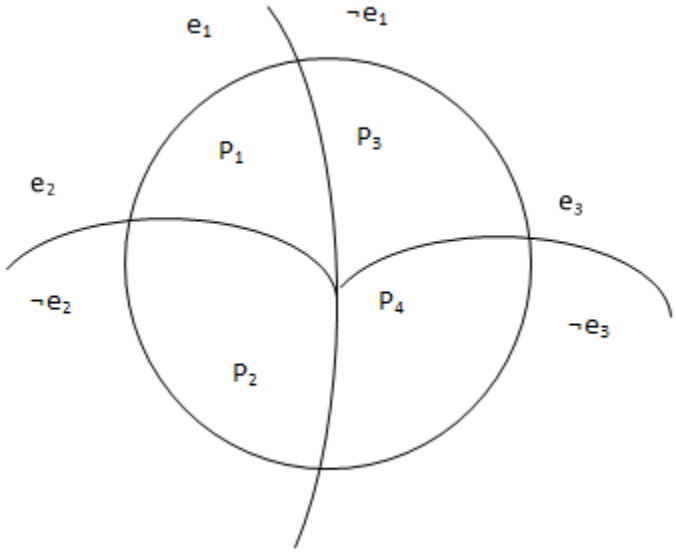

Figure 4: The Assignment of Event Variables to Possible Worlds

based on the same semantics, the so called possible worlds. Following the representation model proposed in [4] for probabilistic relations, the content of the annotations of the tuples are logical expressions similar to uncertain relations. That is, every tuple in every relation instance is associated with a logical expression over event variables in the case of probabilistic relations, and over tuple identifiers, variables, in the case of uncertain relations. Our next step is to find a representation model for probabilistic and uncertain relations which leads to efficient query processing techniques.

2.3.4 Semiring Model

The aforementioned different representation models proposed extending the standard relational data model with annotations to capture either uncertainty or probability. The "semiring model" proposed by Green *et* al. [6] represents annotated relations

which are not only limited to probabilistic and uncertain relations but also relations with bag/multiset, or provenance semantics.

The model is rich and deserves more investigation. In particular, the model is equipped with an algebraic structure to represent the annotation [6] . "Such a structure consists of a set together with one or more binary operations, which are required to satisfy certain axioms [9]." This makes the semiring model practically interesting to explore.

The algebraic structure proposed to capture uncertainty is "commutative semiring". The authors argue that RA operations can be extended strongly identical to the basic algorithms of semiring [6] implying efficient query processing techniques. The model is closed under a subset of RA operations such as projection, selection, union and join which are RA operations of our interest. In what follows, first query evaluation of uncertain data will be discussed and then we will relate it to the semiring model in order to realize all the strengths and weaknesses of the model in terms of query processing.

2.4 Query Evaluation over Uncertain Relations

So far we discussed the issues surrounding representation models of uncertain relations. Now, we focus on issues related to processing queries over such data. First, we begin with the semantics of query evaluation over such data, illustrated in Figure 5.

As it is illustrated in Figure 5, Possible worlds of two uncertain relations, PW(R)

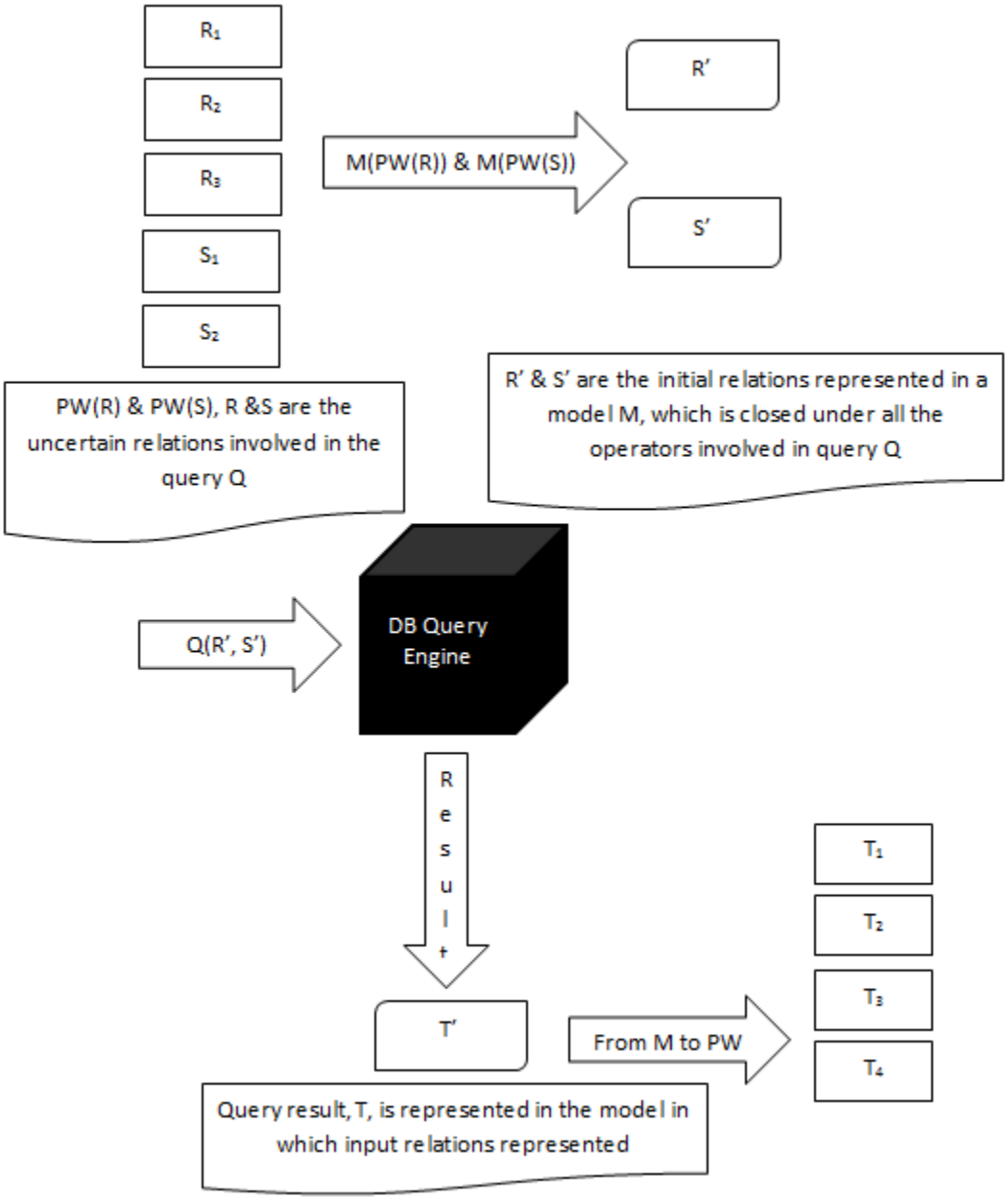

Figure 5: Semantics of Queries Evaluation Over Uncertain Relations

and PW(S), are given. There is a model M which can model uncertainty associated with the initial relations. So we can get two uncertain relations, R' and S', which encode the possible worlds that we started with. The model is also closed under a subset of RA operations involved in the query Q. The closure of the model under that subset guarantees that there is reasonable algorithms for every involved operation so that Q can be applied directly over R' and S' rather than their own possible worlds. It also affirms the fact that query result can be represented by the model M as the initials relations. The reason that DB query engine is demonstrated by a black cube is that the implementation of RA operations are model-dependent. That is, the choice of model has a major influence on the way RA operations are extended. In what follows some terms are introduced and query processing of uncertain relations is studied in detail.

Query evaluation over uncertain data can be discussed based on two different semantics: *intentional* semantics and *extensional* semantics [4]. Intentional semantics is like a validation test that indicates what is expected to get as a query result, by applying the query over every possible world of the involved relations. However, it is not of practical interest due to the exponential number of possible worlds with respect to the number of tuples. Extensional semantics on the other hand is not worried about the semantics and evaluates a query when assured of the query results being correct. The question is how to be assured that the query result is correct? The answer is the closure of the model by which initial relations are represented under RA operations involved in the query.

Indeed, the closure of the model under some RA operations implies that we could have an effective algorithms for those operations such that the query is evaluated over uncertain relations as are, rather than evaluating the queries over a set of possible worlds. As discussed before, this is shown in Figure 5 where we apply query Q(R',S') directly over R' and S' because of the closure of the model M under which the initial relations are represented under all the RA operations involved in query Q.

One may argue that, independent of the kind of manipulation done over the data, the result would be a set of possible worlds and completeness of a model guarantees that this set can be represented by the model. In other words, a complete model is closed under all RA operations implying that there should be a reasonable implementation for every RA operations in the complete model which allows the query to be evaluated directly.

However, in practice neither C-tables nor M^A_{prop} , at top of the hierarchy of working models, is closed under all the RA operations and none of the proposed models are as expressive as the standard relational data model. Indeed, based on the properties of the representation model, RA operations should be extended to ensure that the underlying processing techniques manipulate the annotations properly. Development of such algorithms in practice has been a challenge and difficult to always guarantee.

Based on the discussions in this section, the issues of modeling and query processing are both challenging for uncertain and probabilistic relations. In addition these two issues have to be studied together and balanced. In the following, we study how semiring model is evaluated in terms of query evaluation since it is chosen as our

representation model.

2.4.1 Semiring Model and Query Evaluation

Query evaluation may be done entirely by the query processing module of a DBMS and/or possibly additional processing in the application module. The goal is to extend the modeling and processing capabilities of a conventional relational DBMS to develop a "suitable framework" to manage annotated relations. In the following, we explain how to use the semiring model to contribute to this goal.

Green *et* al. [6] proposed semiring as the algebraic structure to represent and manipulate annotations in annotated relations including uncertain relations and probabilistic relations. They define a subset of RA operations that can be extended efficiently in this model, thanks to the algebraic structure defined. Consequently, the model as a whole can respond to almost all our needs to realize the framework to evaluate queries over uncertain and probabilistic relations.

However, for probabilistic relations, the model follows the definition and algorithm proposed by Fuhr *et* al. [5] in order to derive annotations and compute their associated probabilities which is not an adequate model to meet our goals. Therefore, we apply the model proposed by Dalvi *et* al. [4] in the category of probabilistic relations. In other words, we redress the algorithms proposed by semiring model using Dalvi *et* al. [4] proposal. This leads to the discovery of this fact that not only probabilistic relations and uncertain relations follow the same so called "possible worlds semantics" but also the algorithms involved to manipulate annotation in order to extend RA

operations are the same. This defines the theoretical basis of our work which will be discussed in the next chapter by providing some motivating examples and elaborating the models in detail.

Chapter 3

The Proposed Model

As discussed in Chapter 2, none of the incomplete models in the hierarchy of working models are closed under the join operation, and as such they are not suitable in our context. On the other hand C-table [8] is a complete model and as expressive as M^A_{prop} in the hierarchy of working models, and the RA operations of uncertain relations in semiring model, the representation model of the proposed framework, are extended according to the algorithms proposed in C-table. So in the category of uncertain relations, semiring model is a good choice.

However, we are interested in probabilistic relations defined based on possible worlds semantics also. Semiring model is a representation model of probabilistic relations too, but the RA operations are extended based on the algorithms proposed in [5] which does not follow so-called possible worlds notations. To compensate, we integrate the algorithms proposed by Green *et* al. [6] to semiring model. This manipulation in semiring model leads to an identical semiring extending RA operations of

uncertain relations as well as that of probabilistic relations and contributing to the unification of query evaluation over aforementioned relations.

Once the theoretical basis of the framework was established, we built a running prototype of the proposed framework. In the rest of this chapter, we discuss in detail about the models that are the basis of our framework. The two models named "semiring model" [6] and "probabilistic relations" [4] are explained. Some motivating examples related to each of them are then provided to bring better understanding of the models and motivate our work.

3.1 Provenance Semiring Model

The idea of this model is that "there is a comprehensive representation model that uses semiring of polynomials in order to model different types of annotated relations including uncertain relations, probabilistic relations, bag semantics and how-provenance" [6]. We begin with the definition of the semiring model and explain why the choice of commutative semiring as an algebraic structure can contribute to development of a unified framework for evaluating algebraic queries correctly and efficiently.

3.1.1 Commutative Semiring

A semiring is a set R equipped with two binary operations, + and . , called addition and multiplication [9] and satisfies the following conditions.

1. (R,+,0) is a commutative monoid with the identity element $0 \in R$, such that $\forall$ a, b,c $\in$ R:
   - a+b=b+a
   - a+0=0+a=a
   - a+(b+c)=(a+b)+c
2. (R,.,1) is a monoid with the identity element $1 \in R$, such that $\forall$ a, b,c $\in$ R:
   - a.1=1.a=a
   - a.(b.c)=(a.b).c
3. Multiplication left and right distributes over addition, such that $\forall$ a, b,c $\in$ R:
   - a.(b+c)=(a.b)+(a.c)
   - (a+b).c=(a.c)+(b.c)

Green *e*t al. [6] consider relational algebra calculations for annotated relations as particular cases of the general algorithms of semiring.

3.2 Answering Queries Over Annotated Relations

In standard databases, when a query is posed against a database it is translated into an RA expression, optimized and then executed to produce the query result. This is different in the context of annotated relations in which RA operations are first extended or generalized so that they can manipulate annotations correspondingly.

The efficiency of query processing techniques relies mainly on the algorithms that manipulate the annotations which are in turn dependent on the algebraic structure that define the annotations. In other words, defining algebraic structure for the annotation provides enough flexibility to extend RA operations without being worried about the nature and the content of the annotations. This can also lead to a unified framework to support and evaluate queries in a number of categories illustrated through some examples in the next section.

3.2.1 Motivating Examples

Consider an annotated relation R(A,B,C), shown in Table 12, with different annotation semantics. Last column is Annotation column in general, which is not part of the schema of the relation. For better readability in each semantics we change the name of Annotation accordingly to reflect the true semantics.

A	B	C	Uncertainty
a	b	c	b_1
d	b	e	b_2
f	g	e	b_3

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

A	B	C	Multiplicity
a	b	c	2
d	b	e	5
f	g	e	1

A	B	C	Probability
a	b	c	x
d	b	e	y
f	g	e	z

Table 12: An Instance of R in Different Semantics Categories

Also consider a query q(R) formulated over relation R.

$q(R) = \pi_{AC}((\pi_{AB}(R) \bowtie \pi_{BC}(R)) \cup (\pi_{AC}(R) \bowtie \pi_{BC}(R)))$

The semiring model is closed under selection (σ), projection (π), join ($\bowtie$), and union ($\cup$).

Also note that the query q(R) includes the RA operations under which the model is closed. So, the RA operations are applied directly on the current relation instance involved, without requiring the enumeration of the possible worlds for those categories of semantics and the query result is then generated

Let us review the query result of q(R) where R is an instance of annotated relations in different semantics.

First, we start with an instance of R as an uncertain relation. The query result is illustrated in Table 13. In this category, annotations capture uncertainty which are tuple identifiers or variables in the simplest form or any arbitrary logical expression in more complex form. In addition, the way in which the annotations are manipulated provides enough expressiveness to the model so that any possible type of uncertainty introduced in the process of query evaluation can be also captured.

Table 13: Result of q(R) where R is an Uncertain Relation

A	C	Uncertainty
a	c	b_1
a	e	$b_1 \wedge b_2$
d	c	$b_1 \wedge b_2$
d	e	b_2
f	e	b_3

When R is a probabilistic relation, the query result is shown in Table 14. In this category of relations, annotations represent event variables (EV), which could be a simple or a complex one. Let us elaborate more on this: If the input relations

are probabilistic relations, the EVs are always simple as probability values are replaced with EVs to make intensional query evaluation semantics possible. However, as pointed out earlier any set of possible worlds and their associated probabilities can be represented as a probabilistic relation. To capture the correct semantics of possible worlds, complex EVs may be introduced.

Table 14: Result of q(R) where R is a Probabilistic Relation

A	C	Probability
a	c	x
a	e	x$\cap y$
d	c	x$\cap y$
d	e	y
f	e	z

In the category of probabilistic relations, the extended RA operations introduced by Green *e*t al. [6] are not based on possible worlds semantics but rather they use the algorithms proposed in [5]. However, for different reasons as explained earlier including considering probabilistic relations as a specific type of uncertain relations with possible worlds semantics, and not having independence assumption in query evaluation process we follow the algorithms proposed by Dalvi *e*t al. [4] which will be explained in detail later in this chapter.

In the following we will look into the query results of two more categories with bag and how-provenance semantics. Although, their semantics is not the focus of this work, they can be considered as examples to reconfirm the unification obtained by semiring model.

The query result when R is a relation with bag semantics is shown in Table 15.

In this category, due to the semantics annotations represent the multiplicity of every tuple in the relation instance.

Table 15: Result of q(R) where R is a Relation with Bag Semantics

A	C	Multiplicity
a	c	8
a	e	10
d	c	10
d	e	55
f	e	7

The result of query q(R), when R is an instance of how-provenance relation is depicted in Table 16. In such relations, the annotations indicate not only the contributing tuples but also the way in which they contribute to query result. To capture and represent this semantics, semiring of polynomials is introduced as the algebraic structure of the annotations.

Table 16: Result of q(R) where R is a Relation with How-Provenance Semantics

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

Observation: By dividing the content of the annotated relations into "pure data" and "annotation" parts, we note that pure data of the query result in any category is the same. The annotations on the other hand are different and defined based on the semantics of the relations category. This implies that query can be evaluated on "pure data" in a *uniformed* way however, we need to perform the manipulation of

annotations properly. The following section explains how this could be also done in a *uniformed* way.

3.3 Positive Relational Algebra in Semiring Model

In relational data model, tuples are defined as a function t: $U \rightarrow D$, where U is a *finite set of attributes* and D is a *domain of values*. Following [6], we refer to these relations as standard relations and tuples belonging to these relations as U-tuples. To be able to generalize this definition for annotated relations, K-relations in which tuples are tagged with elements of K, the set of all U-tuples are called U-Tup and R as an annotated relation is defined as R: $U - Tup \rightarrow K$ [6]. In K-relations based on the elements of K, some cases may occur where R(t)=0 so to figure out the tuples of every relation its support is defined following [6]: supp(R)=$\{t \mid R(t) \neq 0\}$

By this introduction on K-relations, the positive relational algebra operators are then defined as follows [6]:

Definition: Suppose (K, +, ., 0, 1) is an algebraic structure in which + and . are two binary operators, and 0 and 1 are two distinct elements in K [6]:

empty relation ϕ :U-Tup $\rightarrow K$ such that $\phi(t) = 0$

union R1 and R2 : R1 $\cup R2$:U-Tup$\rightarrow K$ is definded as

$(\mathrm{R}_1 \cup R_2)(t) = R_1(t) + R_2(t)$

projection R : $\pi_v(R)$:V-Tup$\rightarrow K$ for V as a subset of U is defined as

$(\pi_v(R))(t) = \sum_{t=t' on V and R(t') \neq 0} R(t')$

selection R: the selection σ on predicate P maps every U-tuple to either 0 or 1, i.e $\sigma_p(R)$:U $\rightarrow K$ is defined as $(\sigma_p(R))(t) = R(t).P(t)$

natural join R_i :$U_i \rightarrow K$ i=1,2 , $R_1 \bowtie R_2$ is the K-relation over $U_1 \cup U_2$ defined by: $(R_1 \bowtie R_2)(t) = R_1(t_1).R_2(t_2)$ so that $t_1 = t$ on U_1 and $t_2 = t$ on U_2

The above definitions are keys to unify positive relational algebra operators for different categories of annotated relations [6].

For every category we need to define a semiring consisting of an appropriate set of values and two binary operators plus two identity elements. Below are different semirings based on different semantics [6]:

Set semantics, Standard relations: ($\mathbb{B}$, $\vee$, $\wedge$, false, true)

Bag semantics, Multiset category: ($\mathbb{N}$,+, . ,0,1)

Algebra on C-tables, Uncertain relations category: (set of Boolean expressions, $\vee$, $\wedge$, false, true)

Lineage/How-provenance, How-provenance category: ($\mathbb{N}$[X], +, . ,0,1)
First, let us see how positive RA is defined over probabilistic relations following [5] in semiring model [6]:

Algebra on event tables, Probabilistic relations category:$(\rho(\Omega), \cup, \cap, \phi, \Omega)$
However, we do not consider it as an underlying algorithm to evaluate queries over probabilistic relations in order to develop our framework. Because event tables [5], as a representation model for probabilistic relations, do not conforme to possible worlds semantics. So, they cannot contribute to the unified framework which we are looking for.

On the other hand, probabilistic relations in [4] not only follow possible worlds semantics but also the algebra defined to evaluate queries over such relations is totally compatible with the algebra defined in C-tables to evaluate queries over uncertain relations.

Following such algorithms, the independence assumption of the tuples during the query processing which is made often in other representation models for probabilistic relations is not necessary. As the proposed algorithms can capture any possible interdependency introduced during query processing.

By revising semiring model [6] in the category of probabilistic relations and redress the corresponding algorithms by the algorithms proposed in [4] we get the following semiring:

Algebra on EVs, Probabilistic relations category: (set of Boolean expressions, $\vee, \wedge$, false, true)

As expected, this revision made the algebra defined for probabilistic relations and uncertain relation identical. This defines the theoretical basis of the unified framework. In what follows we look into the probabilistic relations proposed by Dalvi *e*t al. [4] and different query evaluation semantics over probabilistic relations.

3.4 Probabilistic Relations Model

In probabilistic databases, each tuple has a probability of belonging to a relation in the database [4]. On the other hand the semantics of such relations is defined

by the notion of possible worlds. It can be implied that probabilistic relations are uncertain relations in which probability values are associated with possible worlds. In what follows, we explain how query evaluation semantics of probabilistic relations proposed by Dalvi *e*t al. [4] can yield correct query result. To define the basis of our framework, we adapted them to be integrated with semiring model [6].

3.4.1 Answering Queries Over Probabilistic Relations

In probabilistic relations, tuples are annotated with probability values [4, 5]. It is often assumed that tuples in a relation instance are independent which is true for base relations that is, tuples stored in database relations. However, tuples in a query result may not be independent as lots of interdependency amongst tuples during query evaluation process may occur making this assumption invalid. In what follows, we first show how a query is evaluated in semiring model when the input relations are probabilistic through an example. We then evaluate the same query following different semantics proposed in probabilistic relations. This will justify the revision of the algorithms to evaluate queries over probabilistic relations in semiring model [6] based on the algorithms proposed by Dalvi *e*t al. [4].

3.4.2 Motivating Examples

Consider database D in Table 17 consisting of two probabilistic relations S and T. The query $q(\mathrm{S}^p, T^p)$=$\pi_D(\mathrm{S}^p \bowtie_{B=C} T^p)$.

Table 17: A Probabilistic Database D=$\{S^p, T^p\}$[6]

Identifier	A	B	Pr
s_1	m	1	0.8
s_2	n	1	0.5

Identifier	C	D	Pr
t_1	1	p	0.6

Query Evaluation in Semiring Model

In the semiring model, RA operations for probabilistic relations are extended based on the algorithms proposed for event tables [5]. Following such algorithms the result of q(S^p, T^p) is shown in Table 18.

Table 18: $S^p \bowtie_{B=C} T^p$, $\pi_D(S^p \bowtie_{B=C} T^p)$ In Semiring Model

A	B	C	D	Annotation	Pr
m	1	1	p	$s_1 \cap t_1$	$0.8 * 0.6 = 0.48$
n	1	1	p	$s_2 \cap t_1$	$0.5 * 0.6 = 0.3$

D	Annotation	Pr
p	$(s_1 \cap t_1) \cup (s_2 \cap t_1)$	0.48+0.3-0.14=0.64

Query Evaluation Following Possible Worlds Semantics

As pointed out earlier, in probabilistic relations proposed in [4], possible worlds is considered as semantics basis. In what follows, we show the query result while this semantics is followed. To do so, we begin with enumerating all the possible worlds of database D in Table 17 and computing their associated probability values shown in Table 19. We then evaluate q(S^p, T^p).

Query result obtained following possible worlds semantics is depicted in Table 20.

Note that the probabilities of the Tuples in table 20 are computed based on the

Table 19: Possible Worlds and their Associated Probabilities

Possible instances	Probability
$\mathrm{D}_1 = \{s_1, s_2, t_1\}$	$0.8 * 0.5 * 0.6 = 0.24$
$\mathrm{D}_2 = \{s_1, t_1\}$	$0.8 * (1 - 0.5) * 0.6 = 0.24$
$\mathrm{D}_3 = \{s_2, t_1\}$	$(1 - 0.8) * 0.5 * 0.6 = 0.06$
$\mathrm{D}_4 = \{s_1, s_2\}$	$0.8 * 0.5 * (1 - 0.6) = 0.16$
$\mathrm{D}_5 = \{s_1\}$	$0.8 * (1 - 0.5) * (1 - 0.6) = 0.16$
$\mathrm{D}_6 = \{s_2\}$	$(1 - 0.8) * 0.5 * (1 - 0.6) = 0.04$
$\mathrm{D}_7 = \{t_1\}$	$(1 - 0.8) * (1 - 0.5) * 0.6 = 0.06$
$\mathrm{D}_8 = \{\}$	$(1 - 0.8) * (1 - 0.5) * (1 - 0.6) = 0.04$

Table 20: $S^p \bowtie_{B=C} T^p$, $\pi_D(\mathrm{S}^p \bowtie_{B=C} T^p)$ In Possible Worlds Semantics

A	B	C	D	Pr
m	1	1	p	$\mathrm{P}(\mathrm{D}_1) + P(D_2) = 0.48$
n	1	1	p	$\mathrm{P}(\mathrm{D}_1) + P(D_3) = 0.3$

D	Pr
p	$\mathrm{P}(\mathrm{D}_1) + P(D_2) + P(D_3) = 0.54$

summation of the probability values of those possible worlds contributing to that tuple in the query result.

Observation: The query results conforming to the proposed algorithms in *Semiring Model* and following *Possible Worlds Semantics* are shown in Table 18 and Table 20 respectively. It turns out that these two results are not identical. The result obtained following possible worlds semantics is considered to be correct as it is true semantics of such relations. In what follows, we elaborate on query evaluation semantics in probabilistic relations.

3.4.3 Extensional Semantics of Query Evaluation

In probabilistic relations, tuples are associated with probability values. To evaluate queries over such relations, RA operations should be extended/generalized to manipulate probability values accordingly. Figure 6 shows how probability values are manipulated by applying RA operators. The superscript " e " is used to refer to RA operations in "extensional semantics evaluation".

$$
\begin{array}{lcl}
Pr_R(t) & = & Pr(e_R(t)) \\
Pr_{\sigma^e_c(\mathcal{P})}(t) & = & \begin{cases} Pr_{\mathcal{P}}(t) & \text{if } c(t) \text{ is true} \\ 0 & \text{if } c(t) \text{ is false} \end{cases} \\
Pr_{\Pi^e_{\bar{A}}(\mathcal{P})}(t) & = & 1 - \prod_{t' : \Pi_{\bar{A}}(t') = t} (1 - Pr_{\mathcal{P}}(t')) \\
Pr_{\mathcal{P} \times^e \mathcal{P}'}(t, t') & = & Pr_{\mathcal{P}}(t) \times Pr_{\mathcal{P}'}(t')
\end{array}
$$

Figure 6: Extensional Semantics [4]

Let us consider query q(S^p, T^p) introduced before, over the database D shown in

Table 17. Table 21 shows the query result following the extensional semantics.

Table 21: $S^p \bowtie_{B=C} T^p$, $\pi_D(\mathrm{S}^p \bowtie_{B=C} T^p)$ In Extensional Semantics

A	B	C	D	Pr
m	1	1	p	0.8*0.6=0.48
n	1	1	p	0.5*0.6=0.3

D	Pr
p	(1-(1-0.48)*(1-0.3))=0.64

We observe that the query result in this semantics is not identical to the result obtained following possible worlds semantics which is considered as the correct result. It may be inferred that extensional query evaluation semantics cannot guarantee the correct result. We analyze this semantics later in this chapter.

3.4.4 Intensional Semantics of Query Evaluation

Intensional semantics is a systematic way conforming to possible worlds semantics to evaluate queries over probabilistic relations. In order to apply such semantics, probability values of tuples are replaced with EVs which are then manipulated in the process of query evaluation accordingly. Interestingly the algebra proposed by Dalvi *e*t al. [4] to manipulate EVs shown in Figure 7 is identical to algebra proposed in C-table [8] to manipulate logical expressions associated with tuples of uncertain relations. In Figure 7, superscript " i " is used to refer to RA operations in "intensional semantics evaluation".

Let us evaluate query q(S^p, T^p) introduced before, over D shown in Table 17. To do so, probability values first are replaced with EVs shown in Table 22. EVs are then manipulated based on the algebra defined in Figure 7 shown in Table 23. Finally, to

$$
\begin{array}{lcl}
e_{\sigma_c^i(\mathcal{P})}(t) & = & \begin{cases} e_{\mathcal{P}}(t) & \text{if } c(t) \text{ is true} \\ \bot & \text{if } c(t) \text{ is false} \end{cases} \\
e_{\Pi_{\bar{A}}^i(\mathcal{P})}(t) & = & \bigvee_{t':\Pi_{\bar{A}}(t')=t} e_{\mathcal{P}}(t') \\
e_{\mathcal{P}\times^i\mathcal{P}'}(t,t') & = & e_{\mathcal{P}}(t) \wedge e_{\mathcal{P}'}(t')
\end{array}
$$

Figure 7: Intensional Semantics [4]

compute the real probability values associated with tuples of the query result truth assignment tables of all EVs involved is necessary. Let us see the steps as following:

First Step: Probability values are replaced with EVs.

Table 22: A Probabilistic Database $D = \{S^p, T^p\}$, with tuples being annotated with EV,

A	B	EV
m	1	e_1
n	1	e_2

C	D	EV
1	p	e_3

Second Step: EVs are manipulated based on the algebra defined in Figure 7.

Table 23: $S^p \bowtie_{B=C} T^p$, $\pi_D(\mathrm{S}^p \bowtie_{B=C} T^p)$ In Intensional Semantics

Identifier	A	B	C	D	EV
t_1	m	1	1	p	$(e_1 \wedge e_3)$
t_2	n	1	1	p	$(e_2 \wedge e_3)$

Identifier	D	EV
r_1	P	$(e_1 \wedge e_3) \vee (e_2 \wedge e_3)$

Third Step: The truth assignment table of all EVs involved in the query result is necessary shown in Table 24. In addition, the probability of each row of the table based on the probability values of EVs is computed.

Fourth Step: Finally the probability values associated with EVs of Table 23 are computed. Every tuple is associated with EV, and in some rows of truth assignment table EV is evaluated to TRUE. In fact, to compute the probability of each tuple,

Table 24: Truth Assignment Table

Identifier	e_1	e_2	e_3	Pr
r_1	0	0	0	(1-0.8)*(1-0.5)*(1-0.6)=0.04
r_2	0	0	1	(1-0.8)*(1-0.5)*(0.6)=0.06
r_3	0	1	0	(1-0.8)*(0.5)*(1-0.6)=0.04
r_4	0	1	1	$(1-0.8)*(0.5)*(0.6)=0.06$
r_5	1	0	0	(0.8)*(1-0.5)*(1-0.6)=0.16
r_6	1	0	1	$(0.8)*(1-0.5)*(0.6)=0.24$
r_7	1	1	0	(0.8)*(0.5)*(1-0.6)=0.16
r_8	1	1	1	$(0.8)*(0.5)*(0.6)=0.24$

we add up the probability values of truth assignment table's rows in which EV is evaluated to TRUE. As an example, tuple t_1 in Table 23 is annotated with EV=$(e_1 \wedge e_3)$. This EV expression is evaluated to TRUE in two rows of truth assignment table shown in Table 24 that are r_6 and r_8. That's why to compute the probability value of t_1 in Table 25 probability values of these two rows are added up.

Table 25: Final Query Result In Intensional Semantics

Identifier	A	B	C	D	Pr
t_1	m	1	1	p	$Pr(r_6)+Pr(r_8)=0.48$
t_2	n	1	1	p	$Pr(r_4)+Pr(r_8)=0.3$

Identifier	D	Pr
r_1	P	$Pr(r_4)+Pr(r_6)+Pr(r_8)=0.54$

As it is observed, the query result is identical to the query result obtained following possible worlds semantics as it is expected. Right now we have better sight of different query evaluation semantics and can make better conclusion.

Discussion: Possible worlds semantics is the true semantics of probabilistic databases, however, it is not practical due to the exponential number of the possible

worlds in the number of tuples in the database. Intensional semantics evaluation, on the other hand [4] is a systematic way to evaluate queries correctly, however, to compute the real probability values associated with the tuples of the query result, we rely on truth assignment table of EVs. The number of EVs grows exponentially in the number of possible worlds. The number of possible worlds grows in turn exponentially in the number of tuples. Does this imply that in intensional semantics we are also struggling the same problem related to the exponential growth of possible worlds?

Let us name the strong points of intensional semantics evaluation. First of all, this approach provides a systematic way to evaluate queries over probabilistic relations based on possible worlds semantics as they are, without requiring enumeration of all possible worlds. Second, the algebra proposed in [4] which is the basis in our framework to redress RA operations algorithms of probabilistic relations in semiring model [6] is powerful enough so that any possible introduced interdependency occurred in the process of query evaluation will be captured; no independence assumption made. Third, in real life applications the number of possible worlds is far less than the number of tuples providing some hopes that for a range of applications this approach is sufficient. A valid question is that are we able to do better as intensional semantics evaluation is not the ultimate solution? To answer, we need to have some more analysis on extensional semantics evaluation presented in the following as this is the only efficient way to evaluate queries over probabilistic relations.

3.4.5 Analysis of Extensional Query Evaluation

One thing is certain: extensional query evaluation does not always yield the correct result as it is defined based on independence assumption which is not always the case. On the other hand, there might be a category of queries that do not introduce interdependency amongst the tuple in the process of query evaluation. This opens the discussion regarding the types of the queries. For this purpose we first need to review the query processing in a DBMS and then consider again $q(S^p, T^p)$ over database D shown in Table 17.

Once a query is submitted, **Query Parser** component of DBMS parses the query and generated RA expression. This expression is then optimized in **Query Optimizer** to reduce the cost, time or space, by the help of two subcomponents that are *rule-based query optimizer* and *cost-based query optimizer* [10]. With this introduction, let us review the previous example for that two possible query plans are considered.

(1) $q(S^p, T^p) = \pi_D(S^p \bowtie_{B=C} T^p) \equiv$ (2) $q(S^p, T^p) = \pi_D(\pi_B(S^p) \bowtie_{B=C} T^p)$

The query result of the first query plan following extensional semantics and of the second query plan following the same semantics are shown in Table 26 and Table 27 respectively.

Table 26: $S^p \bowtie_{B=C} T^p$, and Final Result(First Query Plan)

A	B	C	D	Pr
m	1	1	p	0.8*0.6=0.48
n	1	1	p	0.5*0.6=0.3

D	Pr
p	(1-(1-0.48)*(1-0.3))=0.64

Table 27: $\pi_B(S^p)$,$(\pi_B(S^p) \bowtie_{B=C} T^p)$, and Final Result(Second Query plan)

B	Pr
1	(1-(1-0.8)*(1-0.5))=0.9

B	C	D	Pr
1	1	p	0.54

D	Pr
1	0.54

As it is illustrated, they are not identical. Interestingly the query result of the second query plan shown in Table 27 agrees with the one obtained following intensional semantics presented in Table 25. To be able to justify it, we need to know more about queries. In what follows we elaborate different categories of queries proposed in [4].

3.4.6 Types of Queries in Probabilistic Relations

To introduce different types of queries, we first need to talk about the queries in a conventional DBMS. For every query, based on the rules stored in the rule-based optimizer, there are some query plans that all of them yield the correct query result. The purpose of generating so many plans is to find the one, based on the recommendation of the cost-based optimizer, which is more timely and costly effective. In the following we bring the same discussion in the context of probabilistic relations.

In probabilistic relation proposed by Dalvi*et* al. [4], while following intensional semantics over the subset of RA operations under which the model is closed the discussion above is valid theoretically. However, we currently don't have any full fledged DBMS designed to manage probabilistic relations in practice. On the other hand, while following extensional semantics the discussion above is not even theoretically valid. We saw a contradictory example in the previous section in which two different

query plans yield different results.

As pointed out earlier in the discussion section of this chapter, intensional semantics is not the ultimate goal. We still need to improve the query evaluation techniques in the category of probabilistic relations. To be able to get advantage of the efficiency of extensional semantics evaluation, queries are divided into two groups: Safe and unsafe queries.

Safe and Unsafe Queries

Literally, safe queries are those that do not introduce any interdependency amongst tuples in the process of query evaluation process [4]. To formally define them, RA operations are divided into safe operations including selection (σ) and join ($\bowtie$), and unsafe operations including projection (π) and union ($\cup$).

In the simplest form, a query is safe if it only consists of safe operations. However, safe queries are not only limited to this small category [4]. For a query consisting of safe and unsafe operations, there is an algorithm to decide on the type of query. Being safe in this case means that there is at least one safe query plan for the query. Theoretically safe queries of either group can follow extensional semantics and the correct query result guarantees.

For unsafe queries on the other hand there is no safe query plan and they cannot follow extensional semantics. Let us relate the queries' type to our framework.

In our framework, we cannot rely on rule-based optimizer of PostgreSQL, the DBMS on top of that our framework is built, to generate query plans as we are

extending the RA operations supported by our framework. To generate the query plan, some predefined rules based on our grammar are followed yielding a unique query plan. This fact at first place says that if a query includes unsafe operations is considered as unsafe. In other word, because of the limitation of the framework, the exact class of unsafe queries cannot be identified. That is, if a query consists of safe operations is safe and extensional semantics is applied to evaluate queries over such queries. If not, intensional semantics yields the correct query result at the cost of generating truth assignment table.

To sum up this chapter, we chose semiring model proposed by Green *e*t al. [6] as the basis of our framework and revised the proposed algorithms [5]to evaluate queries over probabilistic relations based on the intensional semantics and extensional semantics dictated by query's type [4]. This resulted in a unified way to evaluate queries over uncertain relations and probabilistic relations and very efficient way to evaluate safe queries over probabilistic relations. The details of the framework architecture and its modules are provided in the next chapter.

Chapter 4

Architecture Design of the Framework

In this chapter we present the architecture design and the modules of the proposed framework, built on top of PostgreSQL following a "light weight" approach. We begin with a general review of query processing architecture in a typical relational DBMS. Then, we introduce the modules of our framework and explain how they interact with each other and with the PostgreSQL engine.

4.1 Query Processing Architecture

The architecture of a typical DBMS is illustrated in Figure 8. Given a SQL statement, the main tasks of **SQL Parser** are to first check the syntax of the query and then to resolve the names and references and finally to convert it into the internal format

used by the optimizer. If a query parses, then the internal format of the query is passed for further processing.

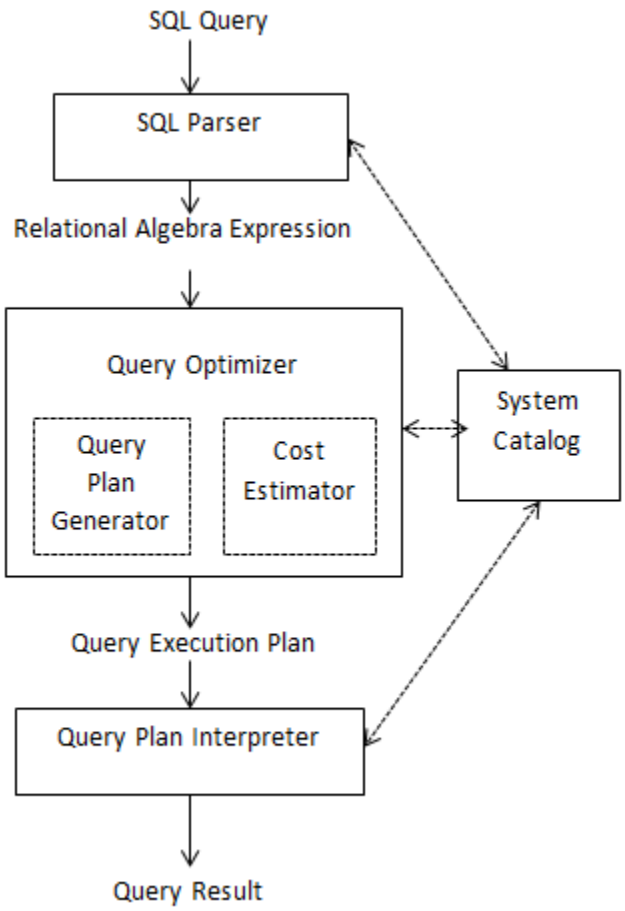

Figure 8: Typical Query Processing Architecture [10]

The job of a **Query Optimizer** is to transform an internal representation into an efficient query plan for executing the query. There is a module called query plan generator inside the optimizer which is a rule-based optimizer generating different query plans according to the stored rules. Cost Estimator on the other hand is a cost-based optimizer which uses the statistics maintained by DBMS to estimate the

cost of the execution of every plan [10]. Query Optimizer can propose a single plan to evaluate RA expression according to the suggestion of cost estimator. The query plan will be executed by **Query Plan Interpreter** and query result will be generated.

4.2 Architecture of the Proposed Framework

The architecture of the proposed framework is illustrated in Figure 9. It consists of three major components: Query Scanner, Query Handler, and Query Processor. They interact with each other and with PostgreSQL DBMS to evaluate queries over annotated relations. In the following, every component will be explained in detail.

4.2.1 Query Validator

RA operations supported by our framework are extended. That is, every operation should manipulate the annotations accordingly. However, PostgreSQL is not aware of the extended RA operations. In other word, we cannot rely on the full strength of PostreSQL to evaluate queries. That is why our framework intervenes in the query evaluation process to guarantee the correctness of the query result.

In addition most probably, queries consists of some subqueries. Every subquery has a RA operation which needs to be taken care of as part of run-time mechanisms. To make it as fast as possible, we decided to define a grammar which is not but almost RA notations. This grammar helps up to extract different tokens of the query such as operands, operators and conditions very efficiently. These tokens contribute

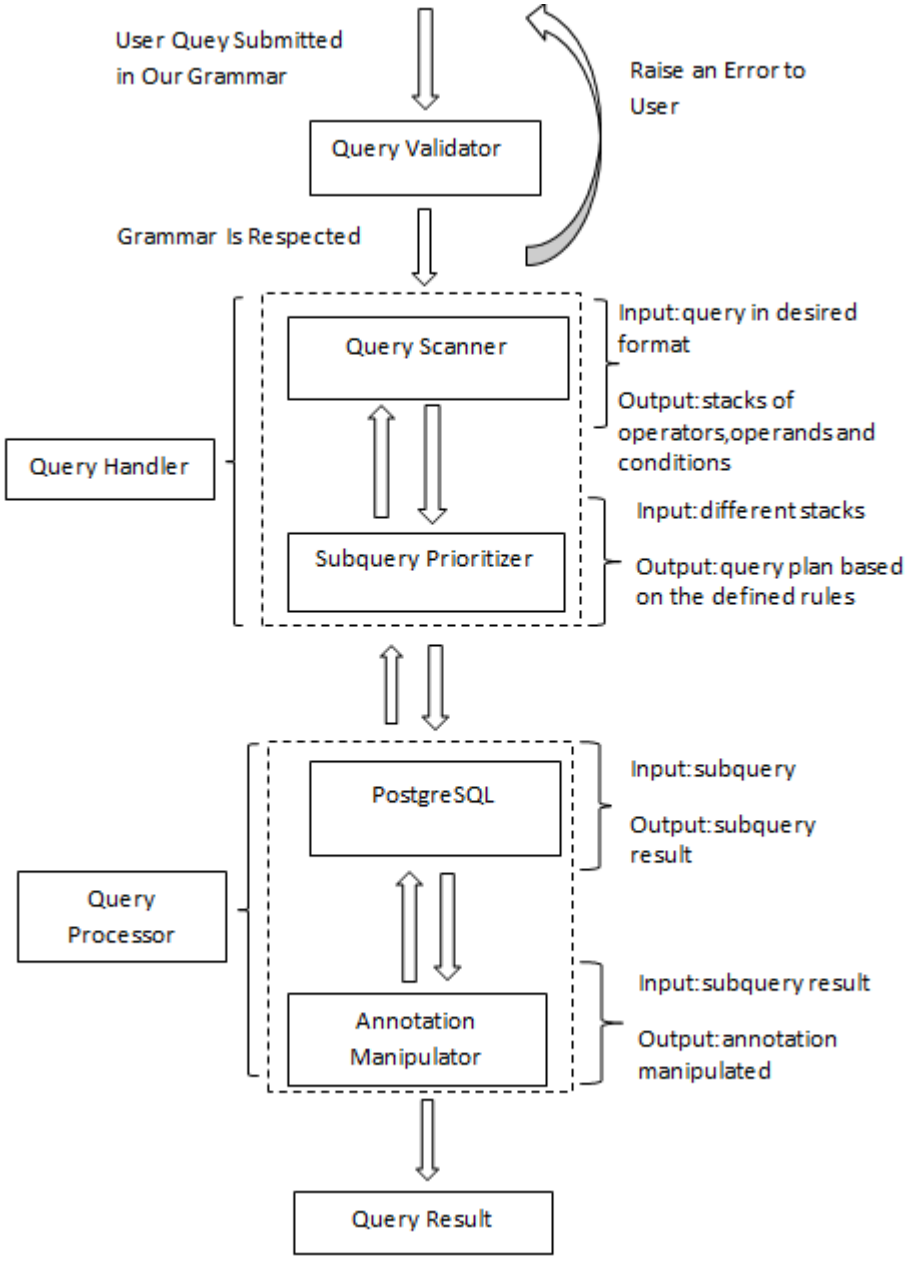

Figure 9: Architecture of the Proposed Framework

to the generation of the query plan also. With this introduction, we can proceed to the tasks defined for Query Validator module of our framework.

Query Validator is to verify the syntactically correctness of the submitted query. First we can list some of the rules of the defined grammar and then we look into a query posed using this grammar.

- There are some reserved letters which stand for different operations.
- The query should be fully parenthesized.
- The order of the execution of the subqueries is based on the inner-most subquery (bracket-off first).
- In case of having the same priority, they are executed from left to right.
- The conditions should be provided accordingly, if any, with respect to the order of the execution of the subqueries.
- There is a naming convention which should be respected while referring to the attributes' names of intermediate results.

example1:

X(a int, b int,c int, Uncertainty text) ,Y(b int, n char, Uncertainty text), and Z(d int, e int, g char, k char, Uncertainty text)

Q(X,Y,Z)= ((p (X j Y)) U (s Z))

X.a=Y.b (join condition)

X_Y_j .a, X_Y_j .b (projection condition)

d,e (selection condition)

As it is shown through the example, grammars rules of the framework varying from the simple parenthesis to the most complex one such as the naming convention should be respected while posing the query . In case there are some syntactical errors, the Query Validator throws an error and asks the user to resubmit the query. Otherwise, the query will be sent to Query Handler as it is shown in Figure 9.

4.2.2 Query Handler

This component is to generate the query plan. It can be compared with Query Optimizer though. However, we follow a very simple rule "bracket off first" to generate the query plan. This component itself consists of two subcomponents described in the following.

Query Scanner

Given a query written syntactically correct and sent from Query Validator, **Query Scanner** scans the query to figure out all its tokens. That is, the query is scanned and divided into different elements including relation names and RA operations. This is the first step toward the generation of the query plan. We continue with the *example1*. Once the Q(X,Y,Z)= ((p (X j Y)) U (s Z)) is scanned, different tokens are divided into three stacks which serve as input for Query Prioritizer. Here is the initial state of these stacks. They might go through several changes during the query evaluation process.

Stack of Operands:{Z,X,Y}

Stack of Operations:{u,s,p,j}

Stack of Conditions:{{d,e},{X_Y_j .a, X_Y_j .b},{ X.a=Y.b}}

Query Prioritizer

The major task of this component is to decide on the order of the execution of the subqueries to guarantee the correct query result. This component interacts directly with Query Scanner From one hand, as the inputs of its major algorithms are elements provided by Query Scanner, and with Query Processor on the other hand which is depicted in Figure 9. Based on the stacks provided by Query Scanner, the query plan of the *example1* turns out to be :

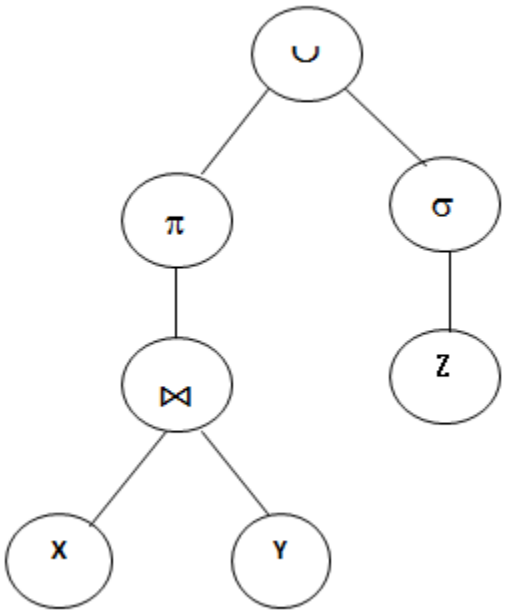

Figure 10: Query plan of *example1*

Now, based on the query plan a subquery to be executed is rewritten as a SQL query and is sent to Query Processor. In *example1*, the first subquery is $select * fromX, Y whereX.a = Y.b$ which is sent to Query Processor.

4.2.3 Query Processor

This is the most important component of our framework as it embodies all the algorithms extending RA operations. To evaluate a query, the framework relies on PostgreSQL and one more component called **Annotation Manipulator** developed to manipulate the annotations accordingly.

As illustrated in Figure 9, this component interacts with Query Handler which provides it a subquery to be executed. Then, PostgreSQL and Annotation Manipulator interact to evaluate it correctly.

Annotation Manipulator

The users are able to choose their desired mode for the query evaluation as our framework supports four different categories of annotated relations. The manipulations of the annotations varies based on the semantics of the relations. Although the choice of semiring provenance as an algebraic structures of the annotations offers a unification to the model. That is, the annotations of different categories are manipulated strongly similar.

Let us continue with the *example1* in which the relations are of uncertain category. To evaluate the first subquery, $select * fromX, Y whereX.a = Y.b$, that should be

executed and is provided by Query Handler: 1) This subquery is sent to PostgreSQL and "pure data " is manipulated. 2) The result is then sent to Annotation Manipulator in which the subroutines of uncertain category are called in order to manipulate the "annotations" accordingly. 3) The intermediate result is stored under the naming convention of the framework. 4) The name of the table containing the result is sent to Query Handler which causes changes in the stacks maintained by Query Scanner. This process is repeated recursively so that the final query result is generated.

Two first components of the framework Query Validator and Query Handler are generic and "category-free". That is, no matter to which category the relations belong, the algorithms implemented to realize the tasks are identical. However, Annotation Manipulator of Query Processor embodies different subroutines with respect to different categories.

In conclusion, the interaction of the framework's components with each other and PostgreSQL as a DBMS results in a running prototype of the proposed framework which can make a step toward the realization of Annotated Database Management System.

Chapter 5

Experiments and Results

In this chapter, we present the results obtained by posing the queries in our framework. The framework is developed by Perl $< v_5, 14, 2 >$ as a programming language on top of PostgreSQL 9.3 as a DBMS. In order to use the framework, we need to install Perl and PostgrSQL on our machine. There are different IDE (Integrated Development Environment) available for Perl to provide more facilities to run the program easily, however, the program can also be run on any OS (Operating System) terminal.

Our proposed framework was built to evaluate queries over different categories of annotated relations which are: Relations with bag semantics simply called multiset, relations with how-provenance semantics called provenance, uncertain relations and finally probabilistic relations. In every category based on the semantics of annotations, the annotations are manipulated accordingly.

The semiring model is the default representation model of the framework and

this model is closed under a subset of RA operations so our framework can support the same RA operations including two unary operations selection, projection and two binary operations union, join. In the rest of this chapter, some examples in every category are provided and the query results are generated by our framework. Note that for readability in every category, the name of annotation column changes accordingly to reflect the exact semantics.

5.1 Multiset

In the multiset category of annotated relations, annotations reflect the multiplicity of the tuples, in other words bag semantics is followed. Consider the multiset T (A, B, Multiplicity) shown in Table 28:

Table 28: Relation T(Mutiset Category)

A	B	Multiplicity
a	b	6
a	c	10
d	c	14

Let us consider examples of unary operations of selection (σ) and projection (π). By applying unary operations on T, we obtain the following results:

Table 29: The Results of Queries $\sigma_{B=c}(T)$ and $\pi_A(T)$ with Multiset Semantics

A	B	Multiplicity
a	c	10
d	c	14

A	Multiplicity
a	16
d	14

To illustrate the binary operations of union ($\cup$) and join ($\bowtie$), consider the multiset

R(C,D,Multiplicity) shown in Table 30.

Note that the schema of T and R are compatible so the union operation makes sense. When the union operation ($\cup$) is applied, the attributes of the first operand is considered as the schema of the query result.

Table 30: Relation R (Multiset Category)

C	D	Multiplicity
e	f	9
f	h	2
a	c	9
d	c	6

Then we consider the queries $T \cup R$, and $T \bowtie_{A=C} R$. The results are shown in Table 31:

Table 31: The Results of $T \cup R$ and $T \bowtie_{A=C} R$ in Multiset

A	B	Multiplicity
a	b	6
a	c	19
d	c	20
e	f	9
f	h	2

T.A	T.B	R.C	R.D	Multiplicity
a	b	a	c	54
a	c	a	c	90
d	c	d	c	84

We next consider queries with different operations and illustrate details of processing such queries. Note that users will only see the final query result. Consider multiset M (Z,Y,Multiplicity) shown in Table 32 and also consider q(M,T,R) defined below:

$q(M, T, R) = \pi_{(T.A)}(\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R))$

To show the steps of query processing, the query $q(M, T, R)$ is divided into subqueries, $\sigma_{Z=a}(M)$, $T \cup R$, and $\sigma_{Z=a}(M) \bowtie_{Z=A} (T \cup R)$, and the result of each step is

shown in Tables 33, 34, and 35 respectively.

Table 32: Relation M(Multiset Category)

Z	Y	Multiplicity
a	c	4
a	d	8
n	f	6
g	k	9

Step 1: Intermediate result of subquery $\sigma_{Z=a}(M)$ is shown in table below. To be able to generate the query result, all intermediate results under the name convention defined in our framework are stored. However, to not make any confusion we did not bring the names in the following examples of this chapter.

Table 33: The Result of Subquery $\sigma_{Z=a}(M)$ in Multiset

Z	Y	Multiplicity
a	c	4
a	d	8

Step 2: Similarly the intermediate result for the subquery $T \cup R$ is:

Table 34: The Result of Subquery $T \cup R$ in Multiset

A	B	Multiplicity
a	b	6
a	c	19
d	c	20
e	f	9
f	h	2

Step 3: Intermediate result for $\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R)$ is shown in Table 35.

Table 35: The Result of Subquery $\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R)$ in Multiset

M.Z	M.Y	T.A	T.B	Multiplicity
a	c	a	b	24
a	c	a	c	76
a	d	a	b	48
a	d	a	c	152

Step 4: The final query result obtained using the results of subqueries is shown in table below.

Table 36: Final Query Result in Multiset

T.A	Multiplicity
a	300

In the following, we pose some queries over provenance relations as a category of annotated relations.

5.2 Provenance

In this category, annotations are the contributing tuples to the query result in addition to the way they contribute. That is, what is captured is not why-provenance but it is how-provenance. To capture such semantics semiring of polynomials is proposed. By default the tuples of the base relations, the relations stored in the database, do not have any annotation. However, these tuples may contribute to some query results later. For the ease of reference, these tuples are annotated with tuple identifiers. Consider the provenance relation T (A, B, Provenance) shown in Table 37:

Table 37: Relation T(Provenance Category)

A	B	Provenance
a	b	s
a	c	r
d	c	m

The examples of unary operations applied on relation T is shown in Table 38.

Table 38: The Result of $\sigma_{B=c}(T)$ and $\pi_A(T)$ in Provenance

A	B	Provenance
a	c	r
d	c	m

A	Provenance
a	s+r
d	m

To illustrate the binary operations of union ($\cup$) and join ($\bowtie$), consider R(C,D,Provenance) an instance of provenance relation shown in Table 39.

Table 39: Relation R in Provenance

C	D	Provenance
e	f	z
f	h	y
a	c	x
d	c	w

Note that the schema of T and R are compatible so we are able to apply the union ($\cup$) operation. When $\cup$ is applied, the schema of the first relation is considered as the schema of the query result. As it is shown in Table 40, the annotations are of form of polynomials. The addition (+) operation indicates either of the operands contributes to that tuple in the query result. However, multiplication (.) operation indicates the presence of both operands can result to that tuple in the query result. Let up pose the queries $T \cup R$ and $T \bowtie_{A=C} R$ over the database D including R and T. The results are shown in Table 40.

Table 40: The Results of $T \cup R$ and $T \bowtie_{(A=C)} R$ in Provenance

A	B	Provenance
a	b	s
a	c	r+x
d	c	m+w
e	f	z
f	h	y

T.A	T.B	R.C	R.D	Provenance
a	b	a	c	s.x
a	c	a	c	r.x
d	c	d	c	m.w

Let M (Z,Y,Provenance) be the provenance relation shown in Table 41.

Table 41: Relation M (Provenance Category)

Z	Y	Provenance
a	c	u
a	d	q
n	f	v
g	k	n

Consider the query $q(R, T, M) = \pi_{T.A}(\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R))$ over the database D. The query is divided into subqueries and the result of each step is shown respectively. Note that, we apply the same query over an identical set of relations with different semantics. Apart from the annotations, the query results remain the same.

Step 1: The intermediate result for the subquery $\sigma_{Z=a}M$ is shown in Table 42.

Table 42: The Subquery Result $\sigma_{Z=a}(M)$ in Provenance

Z	Y	Provenance
a	c	u
a	d	q

Steps 2 & 3: The results of subqueries $T \cup R$ and $\sigma_{Z=a}(M) \bowtie_{(Z=T.A)} (T \cup R)$ are shown in Tables 43 and 44 respectively.

Table 43: The Result of Subquery $T \cup R$ in Provenance

A	B	Provenance
a	b	s
a	c	r+x
d	c	m+w
e	f	z
f	h	y

Table 44: The Result of Subquery $\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R)$ in Provenance

M.Z	M.Y	T.A	T.B	Provenance
a	c	a	b	u.s
a	c	a	c	u.r+u.x
a	d	a	b	q.s
a	d	a	c	q.r+q.x

Step 4: The final query result obtained using the results of subqueries is shown in Table 45.

Table 45: Final Query Result in Provenance Category

T.A	Provenance
a	2u.x+u.r+q.s+q.r+q.x

5.3 Uncertain relations

In this category, the annotations of the tuples capture the uncertainty associated with the data. In the simplest form when there is no interdependency among the tuples and only the presence of the tuples is uncertain, annotation is a literal which can be interpreted as the tuple's ID. Consider the uncertain relation T (A, B, Uncertainty)

shown in Table 46:

Table 46: Relation T(Uncertain Category)

A	B	Uncertainty
a	b	s
a	c	r
d	c	m

The examples of some queries including unary operations on relation T shown in Table 47.

Table 47: The Results of $\sigma_{B=c}(T)$ and $\pi_A(T)$ in Uncertain Category

A	B	Uncertainty
a	c	r
d	c	m

A	Uncertainty
a	s$\vee r$
d	m

Note that the annotations in this category are manipulated based on the algebra proposed in C-table [8]. To illustrate examples of the queries including the binary operations of union ($\cup$) and join ($\bowtie$), consider R(C,D,Uncertainty) as an uncertain relation shown in Table 48.

Table 48: Relation R (Uncertain Category)

C	D	Uncertainty
e	f	z
f	h	y
a	c	x
d	c	w

Then we consider the queries $T \cup R$ and $T \bowtie_{A=C} R$. The results are shown in Table 49. The compatibility of the relations' schema in union operation is considered. In the result of join operation, to facilitate any referencing to the attributes, the name

of attributes change to reflect the origins of the attributes. That is, the attributes' names after join indicate to which relation the attributes belong.

Table 49: The Results of $T \cup R$ and $T \bowtie_{A=C} R$ Over Uncertain Category

A	B	Uncertainty
a	b	s
a	c	r$\vee x$
d	c	m$\vee w$
e	f	z
f	h	y

T.A	T.B	R.C	R.D	Uncertainty
a	b	a	c	s$\wedge x$
a	c	a	c	r$\wedge x$
d	c	d	c	m$\wedge w$

Let M (Z,Y,Uncertainty) be an uncertain relation shown in Table 50.

Table 50: Relation M(Uncertain Category)

Z	Y	Uncertainty
a	c	u
a	d	q
n	f	v
g	k	n

Let us review the steps of evaluating $q(R, T, M)$ in our framework.

$q(R, T, M) = \pi_{(T.A)}(\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R))$

Step 1: The intermediate result for the subquery $\sigma_{Z=a}M$ is shown in Table 51.

Table 51: The Result of Subquery $\sigma_{Z=a}(M)$ in Uncertain Category

Z	Y	Uncertainty
a	c	u
a	d	q

Steps 2 & 3: Similarly the intermediate results for the subqueries $T \cup R$ and $\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R)$ are shown in Tables 52 and 53 respectively.

Table 52: The Result of the Subquery $T \cup R$ in Uncertain Category

A	B	Uncertainty
a	b	s
a	c	(r$\vee x$)
d	c	(m$\vee w$)
e	f	z
f	h	y

Table 53: The Results of the Subquery$\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R)$ in Uncertain Category

M.Z	M.Y	T.A	T.B	Uncertainty
a	c	a	b	(u$\wedge s$)
a	c	a	c	(u$\wedge r$) $\vee$ ($u \wedge x$)
a	d	a	b	(q$\wedge s$)
a	d	a	c	(q$\wedge r$) $\vee$ ($q \wedge x$)

Step 4: The final query result obtained using the results of subqueries is shown in Table 54.

Table 54: Final Query Result in Uncertain Category

T.A	Uncertainty
a	(u$\wedge s$) $\vee$ (($u \wedge r$) $\vee$ ($u \wedge x$)) $\vee$ ($q \wedge s$) $\vee$ (($q \wedge r$) $\vee$ ($q \wedge x$))

In the process of query evaluation, compared to other categories seen so far, the annotations of uncertain relations get more complex making the interpretation and reasoning about the result pretty difficult. In fact, this is the inherent tension existing in the modeling of uncertain data which was discussed earlier. Incomplete models are very limited in terms of uncertainty that they can capture as well as the RA operations that they support . On the other hand, complete models like C-table [8] based on

that RA operations are extended in the semiring model, the representation model of the framework, are more expressive at the expense of being complex.

5.4 Probabilistic Relations

As the last category of annotated relations, we consider probabilistic relations. Annotations of the relations in this category are the probabilities associated with the tuples. In order to evaluate queries, we follow intentional semantics or extensional semantics based on the type of the query. Safe queries consist of only safe operators that are join ($\bowtie$) and selection (σ) and they can be evaluated following extensional semantics. Unsafe queries on the other hand include at least one of the unsafe operators which are projection (π) and union ($\cup$) and they are evaluated following intensional semantics. We elaborate more on these two semantics through the examples. Consider the probabilistic relation T (A, B, Probability) shown in Table 55. Column EV is randomly generated and added to the relation in order to be used in the process of intensional query evaluation.

Table 55: Probabilistic Relation T before and after adding EV

A	B	Probability
a	b	0.8
a	c	0.4
d	c	0.3

A	B	Probability	EV
a	b	0.8	T_34
a	c	0.4	T_18
d	c	0.3	T_78

Consider the query $q(T) = \pi_A(T)$ which is an unsafe query because of an unsafe operation involved. So extensional semantics evaluation is followed. That is, instead

of manipulating the probability values directly we first maintain the annotations (EVs) in the process of query evaluation. Evs are then converted to real probability values based on truth assignment table. As an example of conversion from EVs to probability values is already provided in Chapter 3 we do not bring truth assignment table but the final result. The result is shown in Table 56.

Table 56: The Result of $\pi_A(T)$ Following Intensional Semantics

A	EV
a	(T_34)$\vee$(T_18)
d	T_78

Identifier	A	Probability
t_1	a	0.78
t_2	d	0.3

Now, let us consider another query $q(T) = \sigma_{B=c}(T)$ belonging to safe queries. The query result is evaluated following extensional semantics which is far easier than intensional semantics. That is, the probability values are manipulated directly without requiring to maintain EVs. The result is shown in Table 57.

Table 57: The Result of $\sigma_{B=c}(T)$ Following Intensional Semantics

A	B	Probability
a	c	0.4
d	c	0.3

As discussed before, notion of possible worlds defines the semantics of probabilistic relations. Once a query is evaluated, we are able to compute the probability values of possible worlds of the query result. To make it clear we enumerate all the possible worlds of Table 56 in addition to their associated probabilities. This is shown in the Table 58. Note that enumerating possible worlds of the query result in addition to the computation of their associated probability values is not part of the framework.

This may be only of theoretical interest.

Table 58: Possible Worlds of $\pi_A(T)$ and their Associated Probabilities

Possible Worlds	Probability
$\{t_1, t_2\}$	$0.78 * 0.3 = 0.234$
$\{t_1\}$	$0.78 * 0.7 = 0.546$
$\{t_2\}$	$0.22 * 0.3 = 0.066$
$\{\}$	$0.22 * 0.7 = 0.154$

To illustrate the binary operations of union ($\cup$) and join ($\bowtie$), consider R(C,D,Probability) as a probabilistic relation shown in Table 59.

Table 59: Relation R before and after adding EV in Probabilistic Category

C	D	Probability
e	f	0.4
f	h	0.3
a	c	0.2
d	c	0.2

C	D	EV
e	f	R_43
f	h	R_56
a	c	R_79
d	c	R_14

We first pose a safe query $q(T, R) = T \bowtie_{A=C} R$ evaluated following extensional semantics. The result is shown in Table 60.

Table 60: The Result of $T \bowtie_{A=C} R$ Following Extensional Semantics

T.A	T.B	R.C	R.D	Probability
a	b	a	c	0.16
a	c	a	c	0.08
d	c	d	c	0.06

Let us pose one more unsafe query $q(T, R) = T \cup R$. The result is shown in Tables 61. As the query is unsafe extensional semantics is followed and EVs are maintained. Finally, based on truth assignment table the EVs are converted to the probability

values. The manipulation of EVs rather than real probability values guarantees that any possible introduced interdependency in the process of query evaluation is captured assuring the correct query result.

Table 61: The Result of $T \cup R$ Following Extensional Semantics

A	B	EV
a	b	T_34
a	c	(T_18$\vee R$_79)
d	c	(T_78$\vee R$_14)
e	f	R_43
f	h	R_56

A	B	Probability
a	b	0.8
a	c	0.44
d	c	0.44
e	f	0.4
f	h	0.3

Consider the probabilistic relation M (Z,Y, Probability) shown in Table 62.

Table 62: Relation M before and after adding EV in Probabilistic Category

Z	Y	Probability
a	c	0.3
a	d	0.2
n	f	0.4
g	k	0.1

Z	Y	EV
a	c	M_19
a	d	M_45
n	f	M_12
g	k	M_76

Now, consider the query $q(T, R, M)$ defined as below. The steps of query evaluation are then illustrated.

$q = \pi_{T.A}(\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R))$

Step 1: The intermediate result for the subquery $\sigma_{Z=a}(M)$ is shown in Table 63.

Table 63: The Result of $\sigma_{Z=a}(M)$ in Probabilistic Category

Z	Y	EV
a	c	M_19
a	d	M_45

Step 2: Similarly the intermediate result for the subquery $T \cup R$ is:

Table 64: The Result of $T \cup R$ in Probabilistic Category

A	B	EV
a	b	T_34
a	c	(T_18$\vee$R_79)
d	c	(T_78$\vee$R_14)
e	f	R_43
f	h	R_56

Step 3: The intermediate result for $\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R))$ is shown in Table 65.

Table 65: The Result of $\sigma_{Z=a}(M) \bowtie_{Z=T.A} (T \cup R)$ in Probabilistic Category

M.Z	M.Y	T.A	T.B	EV
a	c	a	b	(M_19$\wedge$T_34)
a	c	a	c	(M_19$\wedge$(T_18 $\vee$ R_79))
a	d	a	b	(M_45$\wedge$T_34)
a	d	a	c	(M_45$\wedge$(T_18 $\vee$ R_79))

Step 4: The final query result obtained using the results of subqueries is shown in Table 66. The value of the EV attribute in this table 66 is too long to fit in the table, instead it is provided as follows:

ev= $(M\_19 \wedge T\_34) \vee (M\_19 \wedge (T\_18 \vee R\_79)) \vee (M\_45 \wedge T\_34) \vee (M\_45 \wedge (T\_18 \vee R\_79))$

Table 66: Final Query Result in Probabilistic Category

T.A	EV	T.A	Probability
a	ev	a	0.426

5.5 Discussion

We extended RA operations in order to evaluate queries over different categories of annotated relations. We illustrated that by using the proposed framework and following respective query processing algorithms, the query results with different semantics are obtained, all in a unified way. Currently, we are not able to use the Query Optimizer module of the PostgreSQL which plays a major role in the success of query processing algorithms. This is mainly because existing standard Query Optimizing techniques are not applicable in our context in general. Simply because reordering arguments or aggregating certainties are allowed only in limited scenarios. In addition, the query plans are generated by the framework and right now it is able to generate the unique query plan which is not necessarily the best plan in terms of time and space. In the next chapter we conclude our work and discuss possible future works.

Chapter 6

Conclusion and Future Work

This work was motivated by problems and issues surrounding representation models and query evaluation techniques of uncertain and probabilistic relations. In the meanwhile we were concerned about the development of a framework to evaluate queries over such data. As uncertain and probabilistic relations are defined based on the same semantics which is possible worlds, we were looking for a model to represent both relations and to yield efficient query processing techniques. The semiring model [6] as a representation model of annotated relations responds to our needs. The model is equipped with semiring provenance as an algebraic structure to model uncertainty or probability associated with data which contributes to efficient query processing. In fact, all the algorithms involved to extend the RA operations are strongly similar to the basic algorithms of semiring which guarantees efficient query processing techniques.

However, the algorithms proposed to extend the RA operations over probabilistic

relations do not consider the possible worlds semantics, our desired semantics, so they cannot always yield the correct query result. To compensate it, we redressed the algorithms based on the proposed algorithms in [4]. That is, for probabilistic relations the framework follows intensional evaluation semantics for unsafe queries and extensional evaluation semantics for safe queries to assure the correct query result. In other word, the revised semiring model defines the representation model of the framework. Once the theoretical basis of the framework was esablished, we developed a framework to evaluate algebraic queries over the annotated relations. We focused on the category of uncertain and probabilistic relations. As expected, their identical semantics led to the identical algebra to extend the RA operations contributing to the unifying framework to evaluate queries.

From semantics point of view, we realized that the mathematical foundation of uncertainty makes the interpretation and reasoning about queries over uncertain relations pretty difficult. That is, evaluation of some queries is possible but query results are complex in general.

However, this problem for probabilistic relations is way bigger. That is, by following intensional query evaluation the computation of the real probability values depends on the truth assignment table of EVs which is exponential in the number of possible worlds. This implies that, we are not always able to compute the result.

From practical point of view, we found out despite all the difficulties involved in the process of development, if we cannot optimize query plans in some possible circumstances, processing queries cannot be guaranteed. This optimization can be

viewed from two different angles: for probabilistic relations, we need to identify the precise class of queries called "safe queries" which can be evaluated following extensional evaluation semantics. However, for uncertain relations, we need to define some rules allowing us to construct different query plans rather than a unique one because all the rules defined in the rule-based optimizer of a conventional DBMS is not valid in our context due to the semantics of data. In other word, reordering the operations involved in the query or aggregating uncertainties is allowed under specific circumstances that needs to be identified. This work could be extended in the following directions.

In the category of uncertain relations, it is very difficult for users to visualize and infer from the results because the logical expression associated with tuples can grow arbitrary long to the number of the tuples in the database. By simplifying the logical expression that is the same as transforming using the axioms of distributive lattices [6] this problem can be somehow mitigated.

In the category of probabilistic relations, any query including unsafe operation is considered as unsafe query. However, we know that this is not the necessary and sufficient condition for unsafe queries. We need to integrate the algorithms proposed in [4] to decide on type of the query precisely. Although for this purpose we need to be able to generate different query plans, if possible.

For any set of possible worlds and their associated probability values, there is an equivalent probabilistic relation [4]. The RA operations of our framework in the category of probabilistic relations are extended following the algorithms defined in

the probabilistic relations [4]. By adding the conversion component converting a set of possible worlds and their associated probability values to a probabilistic relation, we can extend the framework in the sense that the inputs are not necessarily the relations represented in the semiring model but also they are a set of possible worlds.

There is an integration algorithm proposed in [12] whose inputs are the relations represented in the probabilistic relations. By adding the component which can integrate relations represented in the probabilistic relations, we are also able to evaluate queries over the integrated result of some uncertain sources.

In addition to what discussed, there are some improvements entirely related to the framework such as: the framework lacks a user-friendly interface; Currently the interface is identical to SQL Shell (psql) of the PostgreSQL DBMS. One of the modules of the framework is Query Validator which warns the user and provides some hints in case of any syntactical errors. This module can be improved to provide users with more useful hints and help out the user to figure out the mistakes he/she makes. A user manual containing all useful informations and some sample queries can also be provided.

Bibliography

[1] Charu C Aggarwal and Philip S Yu. A survey of uncertain data algorithms and applications. *Knowledge and Data Engineering, IEEE Transactions on*, 21(5): 609–623, 2009.

[2] Parag Agrawal, Anish Das Sarma, Jeffrey Ullman, and Jennifer Widom. Foundations of Uncertain-Data Integration. In *VLDB 2010*. Stanford. URL `http://ilpubs.stanford.edu:8090/846/`.

[3] Omar Benjelloun, Anish Das Sarma, Chris Hayworth, and Jennifer Widom. An Introduction to ULDBs and the Trio System. Technical Report 2006-7, Stanford InfoLab, 2006. URL `http://ilpubs.stanford.edu:8090/793/`.

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

[5] Norbert Fuhr and Thomas Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Transactions on Information Systems (TOIS)*, 15(1):32–66, 1997.

[6] Todd J. Green, Grigoris Karvounarakis, and Val Tannen. Provenance Semirings. In *Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, pages 31–40, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-685-1. URL `http://doi.acm.org/10.1145/1265530.1265535`.

[7] Jiewen Huang, Lyublena Antova, Christoph Koch, and Dan Olteanu. MayBMS: a probabilistic database management system. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 1071–1074. ACM, 2009.

[8] Tomasz Imieliński and Witold Lipski Jr. Incomplete information in relational databases. *Journal of the ACM (JACM)*, 31(4):761–791, 1984.

[9] Dominique Perrin Jean Brestel. *Theory of Codes*. ACADEMIC PRESS,iNC, 1985.

[10] Philip M. Lewis Michael Kifer, Arthur Bernstein. *Database Systems: An Application Oriented Approach,Complete Version*. Pearson, 2005.

[11] Fabian Panse and Norbert Ritter. Completeness in databases with maybe-tuples. In *Advances in Conceptual Modeling-Challenging Perspectives*, pages 202–211. Springer, 2009.

[12] Fereidoon Sadri. On the foundations of probabilistic information integration.

In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 882–891. ACM, 2012.

[13] Anish Das Sarma, Omar Benjelloun, Alon Y. Halevy, and Jennifer Widom. Working Models for Uncertain Data. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, page 7, 2006. doi: 10.1109/ICDE.2006.174. URL http://dx.doi.org/10.1109/ICDE.2006.174.

[14] Anish Das Sarma, Omar Benjelloun, Alon Y. Halevy, Shubha U. Nabar, and Jennifer Widom. Representing uncertain data: models, properties, and algorithms. *VLDB J.*, 18(5):989–1019, 2009. doi: 10.1007/s00778-009-0147-0. URL http://dx.doi.org/10.1007/s00778-009-0147-0.

[15] Prithviraj Sen, Amol Deshpande, and Lise Getoor. Representing tuple and attribute uncertainty in probabilistic databases. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 507–512. IEEE, 2007.

[16] 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 2008 ACM SIGMOD international conference on Management of data*, pages 1239–1242. ACM, 2008.

Appendix A

Appendix

A.1 Illustration of Query Evaluation

The framework supports four different categories of annotated relations. In the welcome message of the framework, the supported categories and their corresponding digits are listed which is shown in Figure 11.

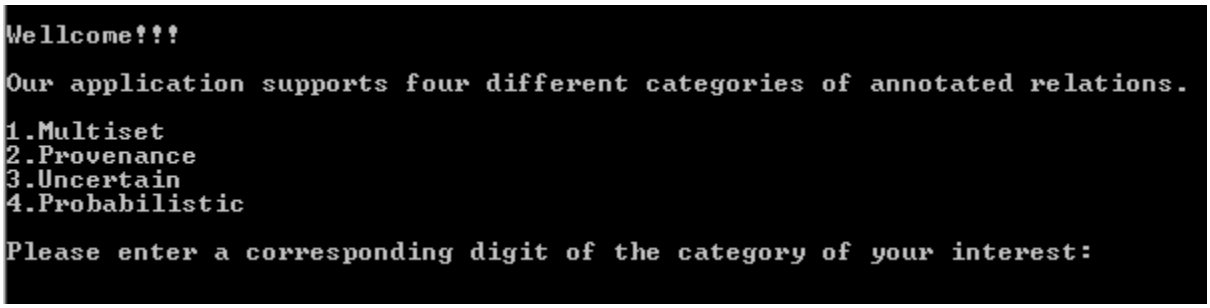

Figure 11: Welcome message of the framework

Once the category is chosen by the user, the query should be submitted following the grammar defined for the framework. In the grammar, some key letters stand

for the RA operations. For instance "s" stands for selection (σ) and "p" stands for projection (π). In addition, the query should be fully parenthesized. An example of the query submitted by the user is shown in Figure 12.

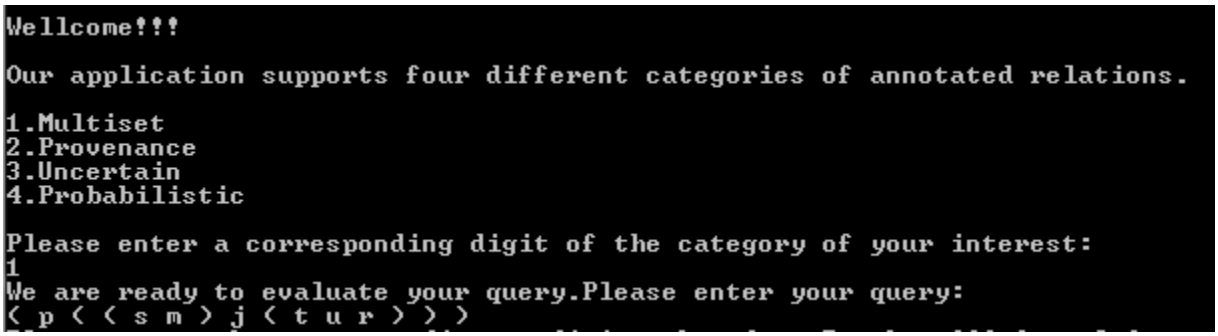

Figure 12: Query Submitted by the User

The corresponding conditions will be then submitted. In the framework, as the intermediate results are stored following the naming convention of the framework, users should be aware of to insert the conditions properly. In Figure 13 the conditions are provided. The intermediate results will be stored to be used to generate the final

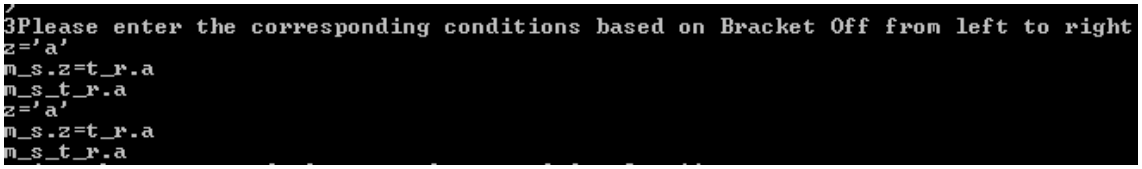

Figure 13: Conditions Submitted by the User

result. However, the users are not aware of them. Once the query is completely evaluated, all the intermediate results will be deleted. To illustrate how the framework keeps track of the intermediate results, they are illustrated. The first subquery which is evaluated is the select statement, select $*$ from M where z='a' . The subquery

result is shown in Figure 14.

```
postgres=# select * from m_s;
 z | y | multiplicity
---+---+--------------
 a | c |            4
 a | d |            8
(2 rows)
```

Figure 14: The Result of Select Statement

The second subquery is union statement, select $*$ from T union select $*$ from R. The subquery result is illustrated in Figure 15.

```
postgres=# select * from t_r;
 a | b | multiplicity
---+---+--------------
 a | c |           19
 e | f |            9
 a | b |            6
 d | c |           20
 f | h |            2
(5 rows)
```

Figure 15: The Result of Union Statement

Respectively, the third subquery is join, select $*$ from (select $*$ from M where Z='a') as First join (select $*$ from T union select $*$ from R) as Second on First.Z=Second.A, which will be evaluated. The result is shown in Figure 16.

```
 m_s_z | m_s_y | multiplicity | t_r_a | t_r_b
-------+-------+--------------+-------+-------
 a     | d     |          152 | a     | c
 a     | c     |           76 | a     | c
 a     | d     |           48 | a     | b
 a     | c     |           24 | a     | b
(4 rows)
```

Figure 16: The Result of join Statement

Finally, the query result will be generated which is shown in Figure 17. The RA

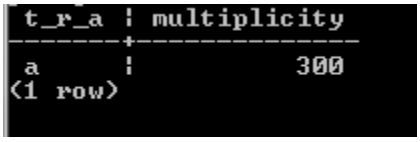

Figure 17: The Final Query Result

operations are extended/generalized in order to manipulate the annotations correspondingly. In other word, in the above example, the bag semantics is followed. We cannot expect to get the same query result by posing the query against PostgreSQL.