A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems

NORBERT FUHR and THOMAS RÖLLEKE
University of Dortmund

We present a probabilistic relational algebra (PRA) which is a generalization of standard relational algebra. In PRA, tuples are assigned probabilistic weights giving the probability that a tuple belongs to a relation. Based on intensional semantics, the tuple weights of the result of a PRA expression always conform to the underlying probabilistic model. We also show for which expressions extensional semantics yields the same results. Furthermore, we discuss complexity issues and indicate possibilities for optimization. With regard to databases, the approach allows for representing imprecise attribute values, whereas for information retrieval, probabilistic document indexing and probabilistic search term weighting can be modeled. We introduce the concept of vague predicates which yield probabilistic weights instead of Boolean values, thus allowing for queries with vague selection conditions. With these features, PRA implements uncertainty and vagueness in combination with the relational model.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*retrieval models*; H.2.1 [**Database Management**]: Logical Design—*data models*

General Terms: Theory

Additional Key Words and Phrases: Hypertext retrieval, imprecise data, logical retrieval model, probabilistic retrieval, relational data model, uncertain data, vague predicates

1. INTRODUCTION

Imprecision in databases is a topic which is getting growing attention. New applications of database management systems (DBMS), like technical or scientific databases, cannot be handled properly without caring for the intrinsic imprecision of the data. Especially for the integration of information retrieval (IR) and database systems, methods for dealing with uncertainty applied in IR have to be included in an integrated system as well.

Authors' address: Universitaet Dortmund, Informatik 6, 44221 Dortmund, Germany; email: {fuhr; roelleke}@LS6.informatik.uni-dortmund.de.

For this reason, there is a need for data models which can cope with uncertainty.

In the logical view on databases, computing the answer to a query q from a database means to find all objects o for which the logical formula $q \leftarrow o$ is true. If one would take the same approach to document retrieval, then a document d should be retrieved in response to a query if $q \leftarrow d$ can be shown to be true. In fact, this is exactly what Boolean retrieval does. However, since IR has to deal with vagueness and imprecision, this approach is not adequate. For this reason, it is argued by van Rijsbergen [1986] that IR should be regarded as an uncertain inference process instead. By using probability theory as basis, van Rijsbergen views document retrieval as being equivalent to computing the probability $P(q \leftarrow d)$ for a document d.

Comparing the two types of inference, one can see that uncertain inference used in IR is just a generalization of the inference mechanism employed in database models. So an integration of IR and databases on the logical level seems to be feasible. In order to arrive at a model which can both be implemented and which is applicable in practice, one has to take a data model from the database field and generalize it such that it also comprises probabilistic inference.

In this article, we present a probabilistic relational model. The basic idea of this model is to assign probabilistic weights to tuples, where the weight of a tuple gives the probability that the tuple belongs to the relation. There is a twofold benefit from this approach. First, uncertain data can be stored in the database. Second, the relation computed as an answer to a query reflects the underlying uncertainty of each tuple in its weight; these tuples can be ranked according to decreasing weights, thus yielding the most certain answers at the top of the list.

The new model is a generalization of the standard relational model. We regard probabilistic relations as generalizations of ordinary (deterministic) relations. In our model, deterministic relations are treated as relations allowing only binary tuple weights (0 or 1), whereas in probabilistic relations, a tuple weight may take any value between 0 and 1. By redefining the basic operators of relational algebra in order to cope with the weights (and their probabilistic interpretation), the laws of relational algebra remain valid, and we get a probabilistic relational algebra (PRA). This way, we can use the full power of relational algebra. If a database contains deterministic relations only, our model yields the same result as standard relational algebra (RA). However, in the presence of imprecise data, the answer to a query may contain tuples with nonbinary weights. If ranking is applied, the "certain" tuples will come out first, followed by the uncertain answers.

In order to handle the probabilistic weights in a proper way, our model is based on intensional semantics (see next section). This is achieved by using canonical propositional formulas as a semantics: each tuple of a relation is accompanied by a so-called event expression, and the PRA operators also manipulate these expressions. The issue of associating probabilities with

these expressions is dealt with separately, namely by first assigning probabilities to the expressions occurring in base relations. Then the expressions in derived relations describe their dependence on the probabilities of the base relations. Using this information, the probabilities of derived relations are computed.

In the remainder of this article, we first describe the motivation that leads to our model. Then we present the probabilistic algebra by giving the basic definitions, followed by a description of the relational operators. In Section 4, we discuss the process of computing the tuple weights, its efficiency, and possible optimizations. Section 5 shows how imprecise attribute values are handled by the probabilistic algebra. As an extension to the basic algebra, vague predicates are presented in Section 6. Implementation issues are described in Section 7. The relationship of our approach to current IR systems and other approaches for imprecision in databases and for the integration of IR and database systems are discussed in Section 8.

2. MOTIVATION

One of the major applications for uncertainty handling in a DBMS is the integration of database and IR systems.

Current commercial IR systems offer very little support for data modeling and query languages of limited expressiveness (and most experimental systems are even worse on these points). As users ask for additional functions, ad hoc extensions of the query language are developed, without a solid theoretical foundation. Furthermore, security issues (when certain users are allowed to view only parts of the database) can hardly be handled by current systems. So it seems to be attractive to apply models developed in the area of database systems to solve these problems. In addition, standalone IR systems have a very limited application range. Usually, retrieval is only one of several tasks which an information system has to perform. For example, in a lending library, a close integration of IR with the processing of the lendings is required (giving the user the information whether or not a book is currently available); for the latter, typical DBMS functions are required.

On the other hand, there are more and more database applications which have to cope with text, too. However, even if a DBMS provides some specific functions for text retrieval, these functions do not take into account the intrinsic uncertainty and vagueness of text retrieval. In the field of IR, several types of models have been developed for coping with this problem, and extensive evaluations have demonstrated the feasability of these approaches. Among them, probabilistic IR models offer the advantage of both solid theoretical foundation [Fuhr 1992b] and good retrieval performance [Harman 1995]. For this reason, a combination of probabilistic IR with a data model like the relational model seems to be a promising goal. But a simple coupling of a standard DBMS with a probabilistic IR system (e.g., such as in Gu et al. [1993]) suffers from the fact that the probabilistic

Index β	DocNo	Term
0.8	1	IR
0.7	1	DB
0.6	2	IR
0.5	3	DB
0.8	3	OOP
0.9	4	IR
0.4	4	AI
0.8	5	DB
0.3	5	OOP

DB β	DocNo
0.7	1
0.5	3
0.8	5

Fig. 1. Relations representing document indexing and retrieval.

weights produced by the IR component cannot be exploited by a DBMS which is based on Boolean logic.

The basic idea of our model is to extend the relational model in such a way that it can handle probabilistic weights required for performing IR. In document indexing, terms are assigned weights with respect to the documents in which they occur (see relation INDEX in Figure 1). There are a number of very effective methods for computing these indexing weights automatically (e.g., Salton and Buckley [1988] and Fuhr and Buckley [1991]). The weights are taken into account in retrieval, where the probability of relevance of a document with respect to the current query is estimated as a function of the indexing weights of the terms involved. As a simple example, consider a query for documents about databases expressed as $\Pi_{\text{DOCNO}}(\sigma_{\text{TERM}=\text{'DB'}}(\text{INDEX}))$, where σ denotes selection and where Π stands for projection. The result is shown as relation DB in Figure 1. It is typical for IR queries that the answer is not just a set of objects: due to the intrinsic uncertainty of IR, the answer should be at least a ranked list of objects. In addition, probabilistic IR models offer an interpretation of the weights that leads to the ranking, namely as estimates of the probability that the object implies the query (which, in turn, can be related to the probability of relevance [Fuhr 1992b]).

Here we have modeled both the indexing weights and the weights of the documents for the query as tuple weights in a relation. Whereas one could imagine other methods of storing indexing weights in a relation (e.g., as an additional attribute), this would not be adequate for modeling the weights of answers to a query. As in the standard relational model, where the answer to a query is always again a relation, we would like to have the same property for our new model. For this reason, we generalize the concept of a relation to probabilistic relations, where each tuple is assigned a weight indicating the probability that the tuple belongs to the relation (ordinary relations can be regarded as a special case where only binary weights may be assigned). So any expression in our probabilistic relational algebra yields a probabilistic relation. This approach contrasts with earlier work on probabilistic data models (e.g., Barbara et al. [1992]) or data models for the integration of IR and database systems (e.g., Schek and

Pistor [1982]), where weights cannot be assigned to a tuple as a whole, since these models are still based on Boolean logic.

Fuhr [1993] describes a probabilistic relational model which is based on extensional semantics. This means that probabilistic weights are attached to tuples; when applying an operator of the relational algebra, the weights of the result tuples are computed as a function of the tuple weights in the argument relation(s). A similar model based on Dempster-Shafer theory with extensional semantics is proposed by Lee [1992]. Both approaches suffer from the fact that the result coincides with the underlying theory for simple relational expressions only. As a simple example, consider a query asking for documents either about DB or IR, but not about both. If we would first compute a relation IR similar to DB from above, the query could be expressed as (DB − IR) ∪ (IR − DB). However, given no additional information, one would have to assume that the two arguments of the union operator are stochastically independent of each other, i.e., for all tuples μ we have

$$P((\mu \in \text{DB} - \text{IR}) \wedge (\mu \in \text{IR} - \text{DB})) = P(\mu \in \text{DB} - \text{IR}) \cdot P(\mu \in \text{IR} - \text{DB}).$$

This is not the case here, since the probability on the left-hand side of this equation equals 0 (the arguments represent disjoint events). In a similar way, the intersection DB ∩ IR cannot be computed as DB − (DB − IR). For some special cases, one might think of a simple solution, e.g., an additional parameter indicating the dependence of the two arguments as described by Lee [1992]. However, it is easy to construct examples where the (in)dependence of the relational arguments varies from tuple to tuple (e.g., see Figure 8), so we need a different approach to solve more general cases properly.

Probabilistic reasoning based on intensional semantics overcomes the difficulties described before. Here the inference process keeps track of the basic events and their dependencies. In PRA, tuple weights are computed from probabilities of the underlying basic events, i.e., tuples of the base relations. For example, assume that db_1 and ir_1 represent the events that document 1 is about DB and IR, respectively. When processing the expression (DB − IR) ∪ (IR − DB), the system would form the corresponding Boolean combination of the underlying basic events, which yields $(db_1 \wedge \overline{ir_1}) \vee (ir_1 \wedge \overline{db_1})$. Given this relationship to the basic events, the correct probability of the result can be computed.

Pearl [1988, pp. 1–14] compares intensional and extensional semantics for models dealing with uncertainty and identifies several flaws of extensional semantics. For the application described here, the problems are due to "improper treatment of correlated sources of evidence" (as shown in the examples above). As further pointed out by Pearl, extensional semantics yields results that are consistent with probability theory for tree-like structures of inference only. Relational algebra expressions often violate the tree structure, so extensional semantics is not appropriate (but see the discussion about simplifying the computation of the probabilities in Section

4). With intensional semantics, however, we can develop a probabilistic relational algebra which always yields correct results (according to probability theory).

3. DESCRIPTION OF THE ALGEBRA

In the following, we first describe the concepts of a probabilistic relation and a probabilistic database, and then we define the operators of the probabilistic relational algebra.

3.1 Basic Definitions

The elementary concepts of domains, attributes, and tuples are the same as in relational algebra. Our formal definitions are similar to the definitions for RA as presented by Vossen [1991], interpreting an attribute value as being a mapping from a tuple into the domain of the corresponding attribute.[1]

Definition 3.1.1. A **domain** D is a finite set of atomic values, for which the predicates $=$ and $\neq$ are defined. For an **ordered domain,** in addition the predicates $<$, $\leq$, $\geq$, and $>$ are defined. Let $\mathfrak{D}$ denote the set of all domains.

Definition 3.1.2. Let $\mathfrak{A}$ denote the set of all **attributes.** The **attribute domain** is a mapping dom : $\mathfrak{A} \rightarrow \mathfrak{D}$. For a set A of attributes, let $dom(A) = \cup_{X \in A} dom(X)$.

Definition 3.1.3. For a set A of attributes, a **tuple over** A is a total and injective mapping $\mu : A \rightarrow dom(A)$, for which the following holds:

$$(\forall X \in A)\ \mu(X) \in dom(X).$$

Let $Tup(A)$ denote the set of all tuples over A, and let $Tup(\emptyset) := \{o\}$.

The injectivity assumption assures that $Tup(A)$ is indeed a set (without duplicate elements). $Tup(\emptyset)$ is defined only in order to simplify the definitions for imprecise attributes (see Section 5), where o denotes a specific, unique tuple that is the only element of $Tup(\emptyset)$.

Since a tuple is an injective mapping, we can use its inverse for mapping between different sets of tuples:

Definition 3.1.4. Let A denote a set of attributes and $\mathcal{P}(A)$ its powerset. Then we define a (nameless) mapping $Tup(A) \times \mathcal{P}(A) \rightarrow \cup_{B \subseteq A} Tup(B)$ denoted as $\mu[B]$, for which the following holds:

$$\mu[B] = \nu \Leftrightarrow \mu \in Tup(A) \wedge \nu \in Tup(B) \wedge B \subseteq A \wedge (\forall X \in B \mu(X) = \nu(X)).$$

[1]The alternate interpretation is to identify attributes with specific positions in a tuple of values. However, this approach would lead to certain difficulties in subsequent definitions.

The schema of a probabilistic relation is the same as that of an ordinary relation. We do not consider keys of relations or other types of integrity constraints here. However, constructs for defining semantic integrity can simply be added to the model presented in this article.

Definition 3.1.5. A **probabilistic relation schema** R is a set of attributes A.

Now we introduce probabilistic concepts. The basic idea of our approach is to associate each tuple of a probabilistic relation with a probabilistic event. A probabilistic relation corresponds to an ordinary relation where the membership of a single tuple in this relation is affected by a probabilistic event. If the event is true, the tuple belongs to the relation; otherwise it does not belong to the relation. For each event, the probability of being true must be given. Special events are $\top$ which always has the value true and $\bot$ always being false.

In terms of the PRA, deterministic relational algebra is a special case where only the events $\top$ and $\bot$ may be assigned to tuples from $Tup(A)$; thus, in RA a relation comprises the set of tuples being assigned $\top$.

We distinguish between basic and complex events. Tuples of the base relations in the database are associated with basic events, which are identified by means of an event key. When new relations are derived from the base relations by means of PRA operators, each tuple in a derived relation depends on certain base tuples from which it was derived. For example, when we form the intersection $A \cap B$ of two base relations A and B, then each result tuple depends on exactly one tuple from A and one tuple from B. In order to express this relationship, we use complex events denoted by event expressions, which are Boolean combinations of basic event keys. Thus, in the last example, each result tuple would be assigned a conjunction of two event keys, one from A and one from B—expressing the fact that the event of the result tuple can be true only if both underlying basic events are true. Generally, the operations of the PRA generate Boolean combinations of event keys. In order to distinguish these combinations from ordinary Boolean expressions, we use the special symbols $\barwedge$, $\barvee$, and $\bar{\neg}$ for the former. Event keys and event expressions form the basis for the computation of the probabilistic tuple weights, as we will show later. In principle, for a probabilistic database only the probabilities for the basic events are given explicitly; from these, all probabilities of complex events can be computed.

Definition 3.1.6. A set of **event keys** E is a set of identifiers, which also contains the special elements $\bot$ (impossible event) and $\top$ (certain event). Furthermore, let $\mathfrak{E}$ denote the set of all possible event keys. The elements in $\mathfrak{E} - \{\bot, \top\}$ are called **probabilistic** event keys, and $\bot$ and $\top$ are **deterministic** event keys.

For a given set E, the set of **event expressions** $ee(E)$ is the set of all Boolean expressions that can be formed with the elements of E and the dyadic operators $\barwedge$, $\barvee$ and the monadic operator $\bar{\neg}$.

In contrast to the relational model, we do not define a probabilistic relation as a set of tuples. Rather, we view a probabilistic relation as a mapping from all possible tuples into a set of event expressions.

Definition 3.1.7. Let $R = A$ denote a probabilistic relation schema. Then a **probabilistic relation** p of type R is a mapping $p : Tup(A) \rightarrow ee(\mathcal{E})$. A **probabilistic base relation** p of type R with respect to a set of event keys E is a mapping $p : Tup(A) \rightarrow E$. Furthermore, let $Rel(A)$ denote the set of all relations of type R with $R = A$.

With regard to an implementation of the PRA and the presentation of probabilistic relations, it is of course sufficient to consider only the "interesting" tuples of a relation p, namely those tuples $\mu \in Tup(A)$ for which $p(\mu) \neq \bot$. For all tuples not considered, it is implicitly assumed that $p(\mu) = \bot$. However, including this condition into the definitions of the operators of the PRA would make some of these definitions rather complex. Therefore, we have chosen the more elegant form of viewing probabilistic relations as mappings.

For specifying the probability of events, we assume that there is a global mapping of all event keys occurring in a database onto probabilities. This approach allows for specifying identical event keys for different tuples, either in the same or in different relations. This way, it is possible to model the probabilistic aspects even of complex objects which cannot be represented by a single tuple.

Definition 3.1.8. A **basic probability assignment** for a set of event keys E is a total mapping $\beta : E \rightarrow [0, 1]$, for which the following holds:

(1) $\beta(\bot) = 0$.
(2) $\beta(\top) = 1$.
(3) $(\forall e \in (E - \{\bot, \top\}))\ 0 < \beta(e) < 1$.

The restriction that event keys different from $\bot$ and $\top$ cannot be assigned probabilities of 0 or 1 simplifies the manipulation of ordinary relations in PRA (see Section 3.3).

With the definitions given so far, we can now specify the concept of a probabilistic database as a set of base relations and the corresponding basic probability assignment.

Definition 3.1.9. Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ denote a finite set of probabilistic relation schemas, where $R_i = A_i$, $1 \leq i \leq k$ and $A_i \neq A_j$ for $i \neq j$. Furthermore, let E denote a set of event keys. Then a **probabilistic relational database over** $\mathcal{R}$ is a tuple $d = (P, E, \beta)$, for which the following holds:

(1) $P = \{p_1, \ldots, p_k\}$ is a set of probabilistic relations such that, for $1 \leq i \leq k$, p_i is a probabilistic base relation of type R_i with respect to E.
(2) β is a basic probability assignment for E.

DocTerm	β	DocNo	Term
DT(1, IR)	0.9	1	IR
DT(2, DB)	0.7	2	DB
DT(3, IR)	0.8	3	IR
DT(3, DB)	0.5	3	DB
DT(4, AI)	0.8	4	AI

DocAu	β	DocNo	AName
DA(1, Bauer)	0.9	1	Bauer
DA(2, Bauer)	0.3	2	Bauer
DA(2, Meier)	0.9	2	Meier
DA(2, Schmidt)	0.8	2	Schmidt
DA(3, Schmidt)	0.7	3	Schmidt
DA(4, Koch)	0.9	4	Koch
DA(4, Bauer)	0.6	4	Bauer

Fig. 2. Example probabilistic relations.

Figure 2 shows an example of a probabilistic database with two relations, where DocTerm gives the weighted index terms for some documents, and DocAu gives the probability that an author is competent in the subjects described in a paper (e.g., in order to distinguish between primary and other authors). As event keys, we use a combination of the relation name and the attribute values here; this eases the understanding of the examples given for the PRA operations below. Of course, in an implementation, an internal ID would be more appropriate for this purpose.

Theoretically, a probabilistic relational database represents a set of ordinary relational databases (with identical schemas) and an additional probability distribution given on these databases. An event expression assigned to a tuple represents the set of ordinary databases containing this tuple. Furthermore, $\bot$ stands for the empty set and $\top$ for the whole set of databases. The probability of an event expression equals the sum of the probabilities of the corresponding ordinary databases.

3.2 Operations

In RA, there are five basic operations: projection, selection, union, difference, and natural join (or, alternatively, cartesian product). Below, we give the corresponding definitions in PRA and illustrate them by means of some examples.

In the relational model, **selection** extracts all tuples from the argument relation fulfilling a certain condition. In PRA, the same effect can be achieved by assigning the event $\bot$ to all tuples not fulfilling this condition. For all other tuples, the event expressions remain unchanged.

Definition 3.2.1. Let $R = A$ be a probabilistic relation schema, r a probabilistic relation of type R, $X \in A$. Furthermore, let $\Theta \in \{<, \leq, >, \geq, =, \neq\}$, if $dom(X)$ is an ordered domain, and $\Theta \in \{=, \neq\}$, otherwise. Then **selection** is a mapping $Rel(A) \rightarrow Rel(A)$ which can take one of two forms:

IRDN	ϱ	DocNo	Term
DT(1, IR)	0.9	1	IR
DT(3, IR)	0.8	3	IR

Fig. 3. Selection: IRDN = $\sigma_{\text{Term}='\text{IR}'}$ (DocTerm).

(1) If $x \in dom(X)$, then the selection $t = \sigma_{X\Theta x}(r)$ of r with respect to $X\ \Theta\ x$ is defined as

$$(\forall \mu \in Tup(A))\ t(\mu) := \begin{cases} r(\mu) & \text{, if } \mu(X)\Theta x \\ \bot & \text{, otherwise.} \end{cases}$$

(2) If $Y \in A$ and X and Y have compatible domains, then the selection $t = \sigma_{X\Theta Y}(r)$ is defined as

$$(\forall \mu \in Tup(A))\ t(\mu) := \begin{cases} r(\mu) & \text{, if } \mu(X)\Theta\mu(Y) \\ \bot & \text{, otherwise.} \end{cases}$$

Figure 3 shows the result of searching for documents about IR by means of the selection $\sigma_{\text{Term}='\text{IR}'}$(DocTerm). (The column headed with ϱ gives the tuple probability in nonbase relations. The computation of these probabilities is described in detail in the following section. In our examples, we compute the ϱ values by assuming that all events are independent.)

Projection is used in RA in order to remove some attributes of the original relation (by means of specifying the remaining attributes as arguments of the projection operator). Two simple examples of this operation in PRA are shown in Figure 4. In the general case, projection may lead to the effect that several tuples of the argument relation are mapped onto a single tuple of the result relation. As an example, consider the projection Π_{Term}(DocTerm) shown in Figure 5, which reduces the relation DocTerm of Figure 2 to the attribute Term only.

In RA, all tuples from DocTerm having the same value for the attribute Term are mapped onto a single tuple, and thus the result consists of the set of index terms occurring in DocTerm. In PRA, we have to specify how the event expression of a result tuple should be computed in this case. From a probabilistic point of view, a tuple should belong to the result relation if at least one of its origin tuples belongs to the argument relation. Thus, we form the disjunction of the corresponding event expressions in this case.

Definition 3.2.2. Let $R = A$ be a probabilistic relation schema, r a probabilistic relation of type R, and $Y \subseteq A$. The **projection** $t = \Pi_Y(r)$ of r onto Y is a mapping $Rel(A) \rightarrow Rel(Y)$ defined as

$$(\forall \mu \in Tup(Y))\ t(\mu) := \bigvee_{\substack{\nu \in Tup(A) \wedge \\ \nu[Y]=\mu}} r(\nu).$$

Now we turn to the set operations **union** and **difference.** In PRA, we form the corresponding Boolean combination of the event expressions. As

IR	ϱ	DocNo
DT(1, IR)	0.9	1
DT(3, IR)	0.8	3

DB	ϱ	DocNo
DT(2, DB)	0.7	2
DT(3, DB)	0.5	3

Fig. 4. Projection: IR = Π_{DocNo} ($\sigma_{\text{Term='IR'}}$ (DocTerm)), DB = Π_{DocNo} ($\sigma_{\text{Term='DB'}}$ (DocTerm)).

Π_{Term}(DocTerm)	ϱ	Term
DT(1, IR) $\veebar$ DT(3, IR)	0.98	IR
DT(2, DB) $\veebar$ DT(3, DB)	0.85	DB
DT(4, AI)	0.80	AI

Fig. 5. Projection with merging of duplicate tuples.

an example, the expression IR $\cup$ DB returns the DocNo of the documents dealing with IR or DB (see Figure 6). The difference IR $-$ DB would return documents dealing with IR, but not with DB.

Definition 3.2.3. Let $R = A$ be a probabilistic relation schema and r, s be probabilistic relations of type R.

(1) The **union** $t = r \cup s$ of r and s is a mapping $Rel(A) \times Rel(A) \rightarrow Rel(A)$ defined as

$$(\forall \mu \in Tup(A))\ t(\mu) := r(\mu) \veebar s(\mu).$$

(2) The **difference** $t = r - s$ of r and s is a mapping $Rel(A) \times Rel(A) \rightarrow Rel(A)$ defined as

$$(\forall \mu \in Tup(A))\ t(\mu) := r(\mu) \wedge \neg s(\mu).$$

The operation **natural join** takes two relations (usually with different schemas) as arguments and yields a new relation, where each tuple is formed from a pair of matching tuples of the argument relations; two tuples are matching if they have the same attribute values for the common attributes. This way, data belonging to the same objects, but spread across different relations due to the normalization process, can be "joined" together again. As an example, the expression DocAu $\bowtie$ DB as shown in Figure 7 gives documents about DB together with their authors. If the argument relations have no common attributes, then **natural join** yields the cartesian product of the argument relations. In PRA, we form the conjunction of the corresponding event expressions. This reflects the interpretation that a tuple is an element of the result relation only if both of its originating tuples belong to their corresponding argument relation.

Definition 3.2.4. Let $R = A$ and $S = B$ be probabilistic relation schemas and r, s be probabilistic relations of type R and S, respectively. Then the **natural join** $t = r \bowtie s$ of r and s is a mapping $Rel(A) \times Rel(B)$

IR ∪ DB	ϱ	DocNo
(DT(1, IR) ⊻ ⊥)	0.9	1
(DT(2, DB) ⊻ ⊥)	0.7	2
(DT(3, IR) ⊻ DT(3, DB))	0.9	3

Fig. 6. Union: IR ∪ DB.

AuDB	β	DocNo	Term	AName
(DT(2, DB) △ DA(2, B))	0.21	2	DB	Bauer
(DT(2, DB) △ DA(2, M))	0.63	2	DB	Meier
(DT(2, DB) △ DA(2, S))	0.56	3	DB	Schmidt
(DT(3, DB) △ DA(3, S))	0.35	3	DB	Schmidt

Fig. 7. AuDB = DocAu ⋈ ($\sigma_{\text{Term}=\text{'DB'}}$ (DocTerm)).

→ $Rel(A \cup B)$ defined as

$$(\forall \mu \in Tup(A \cup B))\ t(\mu) := r(\mu[A]) \triangle s(\mu[B]).$$

The second example for the **join** operation gives an expression which cannot be evaluated correctly with extensional semantics: we search for authors writing about both IR and DB, but possibly in different papers (see Figure 8). Here the event expression for author Schmidt contains two occurrences of the event DA(3,S); this phenomenon violates the implicit independence assumptions underlying approaches based on extensional semantics.

3.3 PRA as a Generalization of Relational Algebra

In contrast to other probabilistic data models (see Section 8) which are only loosely related to the relational data model, PRA is a generalization of relational algebra. More precisely,

(a) PRA contains relational algebra as a special case and
(b) all equivalences from relational algebra also hold for PRA expressions.

With respect to the first point, we can show that relational algebra corresponds to PRA where all relations are deterministic. For that, let the symbol ≡ denote the equivalence of event expressions (according to the laws of Boolean algebra). First, we define the equivalence of ordinary and probabilistic relations: an ordinary relation $\bar{r}$ and a probabilistic relation r are equivalent to each other (denoted by $\bar{r} \cong r$), if and only if

(1) they have identical schemas A and
(2) $(\forall \mu \in Tup(A))\mu \in \bar{r} \Leftrightarrow r(\mu) \equiv \top$.

Then we can show that the following theorem holds:

THEOREM 3.3.1. *For any set* $\{\bar{r}_1, \ldots, \bar{r}_k\}$ *of ordinary relations and any relational algebra expression* $\omega(\bar{r}_1, \ldots, \bar{r}_k)$, *probabilistic relational algebra*

AuIRDB	ϱ	AName
((DT(1, IR) $\triangle$ DA(1, B)) $\triangle$ (DT(2, DB) $\triangle$ DA(2, B)))	0.1701	Bauer
((DT(3, IR) $\triangle$ DA(3, S)) $\triangle$ ((DT(2, DB) $\triangle$ DA(2, S)) $\veebar$ (DT(3, DB) $\triangle$ DA(3, S))))	0.4368	Schmidt

Fig. 8. AuIRDB = Π_{AName} (DocAu $\bowtie$ IR) $\cap$ Π_{AName} (DocAu $\bowtie$ DB).

yields the equivalent result, i.e., if $\bar{r}_1 \cong r_1 \wedge \ldots \wedge \bar{r}_k \cong r_k$, *then* $\omega(\bar{r}_1, \ldots, \bar{r}_k) \cong \omega(r_1, \ldots, r_k)$

The proof of this theorem is given in the appendix.

Concerning the second aspect of generalization (item (b) from above), it should be emphasized that in the case of nondeterministic relations, the event expressions formed for each operator yield a probabilistic interpretation of the operator. In Rölleke [1994], it is shown that all equivalences from relational algebra also hold for PRA. This is due to the fact that these equivalences generate equivalent event expressions.

Since PRA is a generalization of standard relational algebra, we can also exploit the connection between relational algebra and relational calculus: we use the same transformation process from calculus to algebra, but apply the algebraic expression to probabilistic relations. So we have a probabilistic relational calculus which yields probabilistic relations as answers. Thus, PRA forms an implementation of Rijsbergen's view of IR as uncertain inference, since the relational calculus expression for a query can be transformed into an algebra expression, and then the answer can be computed.

4. COMPUTATION OF EVENT PROBABILITIES

The examples from the previous section have shown how the operators from PRA generate new event expressions. So far, we have only defined the probabilities for single event keys. For event expressions, we first need some basic definitions.

Definition 4.1. Let E denote a set of event keys. The **disjunctive normal form** of an event expression $e \in ee(E)$ is its equivalent expression e' of the form $e' = c_1 \veebar \ldots \veebar c_n$, where the c_i are event atoms or conjuncts of event atoms, and an **event atom** is either an event key or a negated event key.

For a conjunct c of event atoms, let $at(c)$ denote the set of its event atoms.

A **minimum conjunct** is a conjunct of event atoms in which each event key occurs at most once.

A **complete conjunct** is a minimum conjunct in which each event key from E occurs exactly once. Let $CC(E)$ denote the set of all complete conjuncts for E.

For example, let $E = \{\top, \bot, e_1, e_2, e_3\}$. Then $c = e_1 \vartriangle e_2 \vartriangle \neg e_2$ is a nonminimum conjunct, with $at(c) = \{e_1, e_2, \neg e_2\}$. Examples of minimum conjuncts are $e_1 \vartriangle e_2$ and $e_1 \vartriangle \neg e_2 \vartriangle e_3$, whereas $\top \vartriangle \neg \bot \vartriangle e_1 \vartriangle e_2 \vartriangle e_3$ and $\top \vartriangle \neg \bot \vartriangle e_1 \vartriangle \neg e_2 \vartriangle e_3$ are complete conjuncts.

For the underlying probabilistic model, $CC(E)$ forms the set of elementary events with respect to the set of basic events E. A single basic event is a set of elementary events, namely all the conjuncts containing this event in unnegated form.

In our view of a probabilistic database as a probability distribution on ordinary databases, each ordinary database is associated with an elementary event. In order to describe the probability distribution, we define a probability measure over $\mathcal{P}(CC(E))$; we have the basic probability assignments given for E, as restriction:

Definition 4.2. Let $d = (P, E, \beta)$ denote a probabilistic relational database. Then a **general probability assignment** is a probability measure ϱ over $\mathcal{P}(CC(E))$ such that $\forall e \in E\ \varrho(e) = \beta(e)$.

Due to the isomorphism of set algebra over $\mathcal{P}(CC(E))$ and Boolean expressions over $ee(E)$, the Boolean expressions also form a Boolean algebra $(ee(E), \bot, \top, \{\vartriangle, \veebar, \neg\})$. For this reason, we can apply the laws from Boolean algebra in order to transform event expressions.

The last definition only sets the frame for the computation of probabilities of event expressions. If we do not want to make any simplifying assumptions, then we have to specify the probability measure by giving probabilities for all complete conjuncts. We call this a complete probability assignment.

For most practical applications, a complete probability assignment will not be feasible. In a database with k different event keys, 2^k probabilities would have to be specified. This is clearly unrealistic for any application of reasonable size. In contrast to that, a rather simple model results if we assume independence of all basic events represented by event keys. Given this independence assumption, the probability of a conjunct can be computed as follows:

Definition 4.3. An **independence probability assignment** is a general probability assignment where the probability of a conjunct c is computed by first transforming c into its equivalent minimum conjunct c' and then evaluating the expression

$$\varrho(c') := \prod_{a \in at(c')} \varrho(a) \quad \text{where}$$

$$\varrho(a) = \begin{cases} \beta(a) & , \text{if } a \in E \\ 1 - \beta(b) & , \text{if } a = \neg b. \end{cases}$$

For general Boolean expressions, we do not have to form the corresponding set of complete conjuncts before we compute the probabilities according

to this definition. Instead, we can also apply the inclusion-exclusion formula [Billingsley 1979, p. 20] by first transforming the Boolean expression e into its equivalent disjunctive normal form $e' = c_1 \veebar \ldots \veebar c_n$ and then computing

$$\varrho(e') = \varrho(c_1 \veebar \cdots \veebar c_n) = \sum_{i=1}^{n} (-1)^{i-1} \sum_{1 \le j_1 < \cdots < j_i \le n} \varrho(c_{j_1} \wedge \cdots \wedge c_{j_i}). \quad (1)$$

The complexity of this evaluation formula is $O(2^n)$. The factor n is determined by the structure of the PRA expression, with the exception of projection, where n also depends on the number of argument tuples being mapped onto the same result tuple.

Complete probability assignment and independence probability assignment only represent the two extremes in a large spectrum of possible (in)dependence assumptions. However, for most applications (like IR), independent events will be sufficient. Furthermore, even when dependence models seem to be more appropriate, often the additional parameters required for these models will hardly be available. It should be emphasized that we are considering information systems for vast amounts of data here. In contrast, most probabilistic inference methods are more suitable for applications like expert systems, where the number of different events is rather limited, and thus it is easier to gather the necessary dependence information. For these reasons, we will only discuss the independence case in the remainder of this section.

For a disjunctive normal form with n conjuncts, the computation of ϱ takes $O(2^n)$ time in the general case. So we should look for possibilities of reducing this computational effort. For this purpose, we consider probabilistic databases with certain restrictions.

Definition 4.4. Let $d = (P, E, \beta)$ with $P = \{p_1, \ldots, p_k\}$ denoting a probabilistic relational database over $\mathcal{R} = \{R_1, \ldots, R_k\}$. d is a **database with unique events** if the following holds:

$$(\forall i, j\ 1 \le i, j \le k)(\forall \mu \in Tup(A_i))$$

$$(\forall \nu \in Tup(A_j))\mu \ne \nu \Rightarrow p_i(\mu) \ne p_j(\nu) \ \vee\ p_i(\mu) = \bot \ \vee\ p_i(\mu) = \top$$

So in a database with unique events, probabilistic event keys are assigned to exactly one tuple in a base relation. Thus, operations combining different base relations will also combine disjoint sets of event keys.

Now let us consider the process of probability computation for this type of database. Besides the possibility of considering dependent events, we have introduced the concept of event expressions mainly in order to cope with possible dependencies of event expressions (caused by identical basic events occurring in these expressions). If no such dependencies exist, then we can compute the event probabilities for the result of an operation

directly from the event probabilities of the relational argument(s), without considering the underlying basic events. This is identical to the computation of probabilities in the case of extensional semantics. Let us call this method *simple evaluation* in the following. So, for PRA expressions where the intensional and extensional semantics approaches would yield the same result, we can eliminate the overhead introduced by event expressions. As an example—like most of the examples from the previous section—the expression $\Pi_{\text{AName}}(\text{DocAu} \bowtie (\text{IR} \cap \text{DB}))$ can be evaluated by means of simple evaluation, whereas the expression

$$\Pi_{\text{AName}}(\text{DocAu} \bowtie \text{IR}) \cap \Pi_{\text{AName}}(\text{DocAu} \bowtie \text{DB})$$

requires the more expensive independence probability assignment.

In general, we can formulate the following theorem:

THEOREM 4.5. *For databases with unique events, the result of a PRA expression can be computed by simple evaluation if and only if for every possible tuple of the result relation, any probabilistic event key occurs at most once in the event expression of the tuple.*

PROOF. Since deterministic events and their Boolean combinations are independent of any other event, their presence does not affect the correctness of simple evaluation. So we only have to consider the probabilistic events here.

Now let us show that simple evaluation yields the correct result in the specified case. If no probabilistic event key occurs more than once in an event expression, then the corresponding probability can be computed iteratively, since the arguments of each Boolean operator are independent of each other (Boolean combinations of different (independent) basic events are also independent). This is exactly what simple evaluation does.

Conversely, suppose that simple evaluation yields correct results, but a probabilistic event key occurs more than once in the event expression. Since we start with basic events, there must be a PRA operation where for some result tuple, two or more of the argument tuples forming this result tuple are derived from the same basic event. Thus, the corresponding events are dependent of each other, and simple evaluation yields an incorrect result. □

So the complexity of our approach is the same as that of one based on extensional semantics as long as both approaches yield the same result. Only for queries where extensional semantics produces wrong results, our approach has a higher complexity. However, since the laws of relational algebra hold for our approach, there also may be an equivalent expression for which simple evaluation is applicable. This has to be determined by the query optimizer.

Further simplifications are possible when we know which relations are deterministic. For example, the last PRA expression from above can be

computed by means of simple evaluation if relation DocAu is deterministic. We are currently investigating the whole area of possible optimizations for the computation of PRA expressions.

5. IMPRECISE ATTRIBUTE VALUES

Our basic model of probabilistic relations also can be extended for coping with imprecise attribute values. For that, we model imprecise attribute values as probability distributions over the corresponding domain. As an example, assume that for some documents the publication year is not known precisely. This fact could be represented by the probabilistic relation DY depicted in Figure 9, showing that document 1 was published either in 1980 or 1981, document 2 certainly in 1990, and document 3 in 1985, 1986, or 1987. Here the events DY(1,1980) and DY(1,1981) are disjoint to each other, in the same way as the three events DY(3,1985), DY(3,1986), and DY(3,1987). On the other hand, it is reasonable to assume that tuples belonging to different document numbers represent independent events.

This example shows that imprecise attribute values are in conflict with the independence assumption underlying the independence probability assignment. For this reason, we modify the independence assumption by treating imprecise attribute values as disjoint events which form exceptions to this assumption. In order to describe the disjointness of this type of event, we introduce the concept of a *disjointness key*. A disjointness key K is a subset of the attributes A of a probabilistic relation, and the attributes in $A - K$ are called imprecise. In the example from above, Year is an imprecise attribute, and tuples with identical values for DocNo represent disjoint events. Thus, we would call DocNo the disjointness key for this probabilistic relation. In general two tuples of a relation represent disjoint events if they have the same values for all attributes of the disjointness key. If the disjointness key is empty, then all tuples of the base relation represent disjoint events.

For the probabilistic relations discussed in the previous sections, there is always $K = A$, i.e., no disjoint tuples. In our examples, the imprecise attributes (belonging to $A - K$) are printed in italics.

For integrating the concept of a disjointness key in our basic model, we consider disjointness keys as a kind of integrity constraint. For a relation with a disjointness key different from the relation scheme, this key induces certain Boolean equations stating the disjointness of events of tuples with identical values of the key attributes.[2]

Definition 5.1. Let $R = A$ denote a probabilistic relation schema. For a **disjointness key** $K \subseteq A$, a probabilistic base relation r of type R yields the following set of Boolean **disjointness equations** $bde(r)$:

[2]One could also consider a disjointness key as being part of the schema of a probabilistic relation. However, this would require the rewriting of most of the definitions given so far. For the sake of clarity of presentation, we choose the solution described here.

DY	β	DocNo	*Year*
DY(1,1980)	0.8	1	1980
DY(1,1981)	0.2	1	1981
$\top$	1.0	2	1990
DY(3,1985)	0.4	3	1985
DY(3,1986)	0.4	3	1986
DY(3,1987)	0.2	3	1987

Fig. 9. A probabilistic relation for the imprecise attribute Year.

—If $K = A$, then $bde(r) := \emptyset$.
—If $K \neq A$, then

$$bde(r) := \{(r(\mu) \triangle r(\nu) = \bot) | \mu, \nu \in Tup(A) \wedge \mu \neq \nu \wedge \mu[K] = \nu[K]\}.$$

At the level of the whole database, we form the union of the Boolean equations from the different relations. Furthermore, the basic probability assignment β has to be defined such that it is not in conflict with disjoint events.

Definition 5.2. Let $\mathcal{R} = \{R_1, \ldots, R_k\}$ denote a finite set of probabilistic relation schemas, where $R_i = A_i$, $1 \leq i \leq k$ and $A_i \neq A_j$ for $i \neq j$. Furthermore, let $\mathcal{K} = \{K_1, \ldots, K_k\}$ denote a set of disjointness keys with $K_i \subseteq A_i$, $1 \leq i \leq k$. Then an **extended probabilistic relational database over** $(\mathcal{R}, \mathcal{K})$ is a tuple $d = (P, B, \beta, E)$, for which the following holds:

(1) $P = \{p_1, \ldots, p_k\}$ is a set of probabilistic base relations such that, for $1 \leq i \leq k$, p_i is a probabilistic base relation of type R_i with respect to E.
(2) $B = \cup_{1 \leq i \leq k} bde(p_i)$.
(3) β is a basic probability assignment for E, such that, for $1 \leq i \leq k$, the following holds:

$$\forall \kappa \in Tup(K_i) \sum_{\substack{\mu \in Tup(A_i) \\ \mu[K_i] = \kappa}} \beta(p(\mu)) \leq 1.$$

For computing the probability of event expressions, we also have to modify the independence probability assignment. The general idea is to use the Boolean equations in order to simplify event expressions. Thus, if a conjunct contains two disjoint event keys, it yields $\bot$. However, we need a special procedure for a conjunct which contains two or more negated event keys which are disjoint. In this case, we have to sum up the probabilities for the unnegated disjoint keys and treat the result as the opposite probability. For example, assume that d_1, d_2, and d_3 are disjoint event keys, and a is an independent event key. Then the probability of the conjunct $a \triangle \bar{d}_1 \triangle \bar{d}_2$ would be computed as

$$\varrho(a \triangle \bar{d}_1 \triangle \bar{d}_2) = \varrho(a) \cdot (1 - (\varrho(d_1 \veebar d_2))) = \varrho(a) \cdot (1 - (\varrho(d_1) + \varrho(d_2))).$$

Definition 5.3. An **extended independence probability assignment** for an extended probabilistic database $d = (P, B, \beta, E)$ is a general probability assignment where the probability of conjuncts containing event atoms only is computed in the following way. Let $c = c_1 \wedge \ldots \wedge c_i$ denote such a conjunct, where $c_1, \ldots c_i$ are the event atoms occurring in c. Then the subsequent steps are performed:

(1) c is transformed into its equivalent minimum conjunct $c' = \bigwedge_{j=1}^{m} s_j$ by using the equations in B such that it does not contain any two positive event keys which are disjoint, that is, $\forall i, j\ 1 \leq i < j \leq m\ (s_i \wedge s_j \neq \bot)$.

(2) c' is further transformed into a conjunct $c'' = \bigwedge_{i=1}^{n} t_i$, with $t_i = \bigwedge_{j=1}^{k_i} \bar{e}_j$, iff $\forall j_1, j_2\ 1 \leq j_1 < j_2 \leq k_i\ (e_{j_1} \wedge e_{j_2} = \bot)$, and t_i is an event atom otherwise. So each t_i is either a conjunction of negative disjoint event keys or an event atom for which there is no disjoint event key in c''. Then we define

$$\varrho(c'') = \varrho(\bigwedge_{i=1}^{n} t_i) := \prod_{i=1}^{n} \varrho(t_i) \quad \text{with}$$

$$\varrho(t_i) = \begin{cases} \beta(t_i) & \text{, if } t_i \text{ is a positive event key} \\ 1 - \beta(e_i) & \text{, if } t_i = \bar{e}_i \text{ is a negated event key} \\ 1 - \sum_{j=1}^{k_i} \beta(e_j) & \text{, if } t_i = \bigwedge_{j=1}^{k_i} \bar{e}_j. \end{cases}$$

For example, the PRA expression $\Pi_{\text{DocNo}}(\sigma_{\text{Year}>1985}(\text{DY}))$ searches for documents published after 1985; as result, we get the event expression (DY(3,1986) $\vee$ DY(3,1987)) for DocNo = 3, with a probability of 0.4 + 0.2 = 0.6.

In comparison with ordinary relations, there is a close relationship between a (nonempty) disjointness key and the key of an ordinary relation. For example, the relation DY from above can be thought of being derived from an ordinary relation BOOK (DocNo, Year, Price, Title) with DocNo as key attribute. When Year is to become an attribute with imprecise values, then Year has to be removed from BOOK, and the new relation DY has to be formed in order to keep relations in third normal form. Since the value of the key attribute(s) determines the values of all other attributes (i.e., the functional dependency DocNo $\rightarrow$ Year in this example), multiple values of a nonkey attribute for a single value of a key attribute correspond to disjoint events.

In our view of a probabilistic database as a probability distribution on a set of ordinary databases, disjoint events represent disjoint subsets of databases. Thus, two tuples with disjoint event keys never cooccur in the same ordinary database. This implies that for each ordinary database, the disjointness key is a key of the relation, since the values of the remaining attributes are determined by the value of the disjointness key. However, in another ordinary database, for the same disjointness key value, the values for the remaining attributes may be different.

DYP	β	DocNo	*Year*	*Price*
DYP(1,1980,20)	0.8	1	1980	20
DYP(1,1981,22)	0.2	1	1981	22
$\top$	1.0	2	1990	19
DYP(3,1985,15)	0.4	3	1985	15
DYP(3,1986,16)	0.3	3	1986	16
DYP(3,1986,17)	0.1	3	1986	17
DYP(3,1987,17)	0.2	3	1987	17

Fig. 10. Imprecise attributes Year and Price.

In principle, due to the first-normal-form condition, a separate probabilistic relation has to be formed for each imprecise attribute from the original relation, e.g., DP (DocNo, Price) if Price has also imprecise values. However, this is true only if the values of Price and Year are stochastically independent. Otherwise, both imprecise attributes have to be included in the same relation, as shown in Figure 10. Here again the disjointness key is {DocNo}, but now we have several pairs of (Year,Price) values for a single value of DocNo, and these pairs correspond to disjoint events. Obviously, this probabilistic relation cannot be decomposed into two separate relations (even if we ignore the probabilities, there is no lossless join decomposition, i.e., a decomposition into two relations R1(DocNo, Year) and R2(DocNo, Price) where the original relation can be reconstructed by means of a join).

With relation DYP, the query for documents published after 1985 would yield the probability 0.6 for DocNo = 3 (as before). Searching for books with a price of at least 17 would give us the probability 0.3 for the same book (and probability 1 for DocNo = 1). A selection with the conjunction of the two conditions by means of the expression $\Pi_{\text{DocNo}}(\sigma_{\text{Year}>1985 \wedge \text{Price} \geq 17}(\text{DYP}))$ would yield the event expression (DYP(3,1986,17) $\veebar$ DYP(3,1987,18)) with a probability of 0.1 + 0.2 = 0.3, since the outcomes of the two selection conditions are not independent of each other. In contrast to models with extensional semantics, the (algebraically equivalent) expression $\Pi_{\text{DocNo}}(\sigma_{\text{Year}>1985}(\sigma_{\text{Price} \geq 17}(\text{DYP})))$ would give us the same result.

Another important application of disjoint events is for query term weighting in text retrieval. So far, we only have considered Boolean combinations of query terms. If we represent the set of query terms as a relation of disjoint events, then we achieve a specific form of query term weighting. For example, a query with the terms DB and IR could be represented as a relation Q1(Term) with the disjointness key {Term} (see Figure 11). Then the expression $\Pi_{\text{DocNo}}(\text{Q1} \bowtie \text{DocTerm})$ would yield the result shown in Figure 12. Since in Q1 the terms represent disjoint events, but in DocTerm they stand for independent events, the probabilities in the result relation are computed as the scalar product of the corresponding weights.

6. VAGUE PREDICATES

In interactive information systems, queries often may be vague. For example, a customer looking for PCs with a price less than U.S. $1000 may be

Q1	β	*Term*
Q1(DB)	0.7	DB
Q1(IR)	0.3	IR

Fig. 11. Query term weighting.

Π_{DocNo}(Q1 $\bowtie$ DocTerm)	ϱ	DocNo
Q1(IR) $\triangle$ DT(1, IR)	$0.3 \cdot 0.9$	1
Q1(DB) $\triangle$ DT(2, DB)	$0.7 \cdot 0.7$	2
Q1(IR) $\triangle$ DT(3, IR) $\veebar$ Q1(DB) $\triangle$ DT(3, DB)	$0.3 \cdot 0.8 + 0.7 \cdot 0.5$	3

Fig. 12. Result of query term weighting.

willing to pay a little more in certain circumstances. In literature databases, a person searching for relevant publications from the past three years may also be interested in a highly relevant paper published four years ago. Different approaches have been proposed for this problem (e.g., Motro [1988], Prade and Testemale [1984], and Zemankova and Kandel [1985].

In Fuhr [1990], we have developed a probabilistic interpretation of vague predicates. In the first example from above, for example, there is a certain probability that the customer will pay a specific price higher than U.S. $1000. This phenomenon can be modeled by means of (1) a probabilistic relation with two attributes containing the values to be compared and (2) the associated probability that a user formulating a query with this vague predicate will accept a specific pair of values. Figure 13 shows some tuples of the relation for the example from above. Given this relation, it is obvious that we could express a vague selection condition like "Price $\lessapprox$ 1000" on a relation R by means of the following PRA expression:

$$R \bowtie \prod_{\text{Price}}(\sigma_{A=1000}(LT)).$$

It is obvious that the relation corresponding to a vague predicate need not be given explicitly—in many cases, the tuple probability could also be specified as a function of the two values to be compared. In Fuhr [1990], it is shown how this type of function can be derived from empirical data.

Following these considerations, we model vague predicates as built-in predicates. In comparison to the standard built-in predicates $=$, $\neq$, $<$, $>$, $\leq$, and $\geq$, vague predicates generate probabilistic events, whereas the standard predicates only yield the values true and false, i.e., $\top$ and $\bot$. In RA, any predicate can be viewed as a (built-in) relation. So vague predicates are a generalization of standard predicates analogously to probabilistic relations as generalizations of ordinary relations.

Definition 6.1. Let D denote a domain. Then a predicate for domain D is a mapping $\Theta : D \times D \rightarrow \mathcal{E}$.

Given this interpretation of predicates, we can reformulate the selection operation to support both standard and vague predicates.

LT

η	β	A	Price
$\hat{E}$	1.0	1000	900
$\hat{E}$	1.0	1000	950
LT(1000, 1000)	0.99	1000	1000
LT(1000, 1050)	0.90	1000	1050
LT(1000, 1100)	0.60	1000	1100

Fig. 13. A probabilistic relation for the vague predicate $\lessapprox$.

Definition 6.2. Let $R = A$ be a probabilistic relation schema, r a probabilistic relation of type R, and $X \in A$. Furthermore, let Θ be a predicate defined for $dom(X)$. Then **general selection** is a mapping $Rel(A) \rightarrow Rel(A)$ which can take one of two forms:

(a) If $x \in dom(X)$, then the selection $t = \sigma_{X\Theta x}(r)$ is defined as

$$(\forall \mu \in Tup(A))\ t(\mu) := r(\mu)\ \vartriangleleft\ \Theta(\mu(X), x).$$

(b) If $Y \in A$, and X and Y have compatible domains, then the selection $t = \sigma_{X\Theta Y}(r)$ is defined as

$$(\forall \mu \in Tup(A))\ t(\mu) := r(\mu)\ \vartriangleleft\ \Theta(\mu(X), \mu(Y)).$$

From these definitions, it can be seen that vague predicates are a natural extension of PRA.

The application of vague predicates is not restricted to numeric attributes. For example, in the case of searching for proper names, a probabilistic interpretation of string similarity (e.g., based on phonetic codes, number of overlapping trigrams, or editing distance) as described in Pfeifer et al. [1995] can be applied as vague predicates. For text retrieval, phrase search also can be interpreted as a vague predicate testing a given phrase for occurrence in a document text (e.g., Croft et al. [1991]);[3] obviously, returning a binary weight as a result of the predicate would not be appropriate for probabilistic retrieval. In multimedia retrieval, the concept of vague predicates seems to be even more important, since many approaches are based on some kind of similarity operation; for example, the QBIC system [Flickner et al. 1995] supports similarity search of images with respect to different features, e.g., texture, contour, or color.

Looking more carefully at the price example from above, we see that the independence assumption does not hold for the events in relation LT (or the events generated by the corresponding vague predicate, respectively), since all these events are dependent on each other. However, this fact does not make the whole approach useless. We only run into problems if we get an event expression which contains two different events from this relation. Below, we will show how to cope with this problem. First, however, we will

[3]Here we assume that the set of possible phrases is too big to allow for preprocessing of phrases such that they could be stored in the same way as other index terms.

take a closer look at the problem of dependence. In general, a PRA expression might contain more than one vague predicate. So the question is this: when are the corresponding events dependent on each other? This problem can be illustrated by means of some examples. Assume that we have a relation PUB(Author, Title, PYear, EYear, Price) about books where PYear denotes the publication year of the first edition and EYear the year of the latest edition. Now consider the following selection conditions for relation PUB containing multiple vague conditions:[4]

(1) Author $\cong$ 'Smith' $\wedge$ PYear $\cong$ 1985.
(2) PYear $\gtrsim$ 1990 $\wedge$ Price $\lesssim$ 30
(3) PYear $\gtrsim$ 1992 $\vee$ PYear $\lesssim$ 1985.
(4) EYear $\gtrsim$ 1992 $\wedge$ PYear $\lesssim$ 1985.
(5) PYear $\gtrsim$ 1990 $\wedge$ PYear $\cong$ EYear.

It is reasonable to assume that, both in expressions (1) and (2), the events for the two vague predicates are independent of each other. In expression (3), we would claim that they are dependent, since we have two vague conditions (although with different predicates) for the same attribute. The same holds for expression (5). In (4), we have different predicates and different attributes, but we would hesitate to say that the events are independent. The reason for this is that the attributes EYear and PYear have the same domain. Since vague predicates bear a lot of semantics, we cannot state the dependence or independence of vague conditions from purely syntactic criteria. Instead, we assume that vague predicates for the same domain are dependent on each other. So certain vague predicates can be applied only to attributes of a specific domain. This means that, for example, in the conditions Price $\lesssim$ 30 and PYear $\lesssim$ 1985, $\lesssim$ is not the same vague predicate. In order to make this more explicit, we could use the domain as index of the vague predicate, e.g., $\lesssim_{\text{Year}}$. On the other hand, we will assume that predicates for different domains are independent of each other. However, we should also note that this general criterion for deciding about the (in)dependence of events generated by vague predicates may not always be appropriate. As a counterexample, in the case of phrase search, any query containing more than one phrase would lead to event expressions containing mutual dependent event keys for the different phrases; thus, it would be more reasonable to assume independence of different events generated by the vague predicate in this case.

In the following, let us assume that we can identify the sets of mutual dependent events, but that there is no additional information (e.g., giving the probabilities of combinations of dependent events). Thus, the independence probability assignment does not allow for computing the probabilities for event expressions containing dependent events. This raises the question regarding which PRA expressions can be processed under this constraint.

[4]Like in RA, a selection with a Boolean combination of atomic criteria can be transformed into an equivalent PRA expression with atomic selection criteria.

For this purpose, we introduce the notion of safety: a PRA expression is called *safe* if, for all possible values of its arguments, no event expression contains more than one event from a set of dependent events. In this case, independence probability assignment yields correct results.

For example, the search for documents with an author name similar to "Maier" by means of the expression

$$\prod_{\text{DocNo}}(\sigma_{\text{AName}\cong\text{'Maier'}}(\text{DocAu}))$$

is unsafe, since the projection may combine different tuples (e.g., a document with the authors "Maier" and "Meier"), where each tuple refers to an event from the relation representing the vague predicate. So the projection operation has to be omitted in order to get a safe expression.

7. IMPLEMENTATION CONSIDERATIONS

PRA is a logical data model. Thus, it specifies only a framework for the implementation of this model. With regard to the typical applications of our model, the physical data model has to be chosen carefully. For example, it would not be wise to store the index terms assigned to a document as a sequence of tuples like for other relations. Having a set of (weighted) terms attached to the document number would be more appropriate. Of course, one would also have indexes or inverted lists as access paths like in typical IR systems. So the logical data model only specifies the behavior of the system at a certain interface.

A similar statement also holds for the handling of imprecise attributes as described in Section 5. Conceptually, a separate relation has to be formed for each imprecise attribute (or combination of stochastically dependent attributes). At the physical level, however, these relations can be stored together such that, for a certain value of the disjointness key, all the values for the different imprecise attributes are stored in the same record.

The two issues addressed above may suggest a different logical model, namely a nested relational model (or NF^2 model—see also Section 8.2). We have described this approach in a separate paper (see Fuhr and Rölleke [1996]). Besides its higher complexity (in comparison to PRA), the major drawback of a probabilistic NF^2 model is that it is no more value based like the relational model: when checking two tuples with set-valued attributes for equality, it is not sufficient to compare the attribute values, and the event keys have to be compared. This situation is similar to object-oriented models where two objects may be equal (having equal attribute values) or identical (having the same object identifiers). Thus, event keys play the role of object identifiers in the probabilistic NF^2 model. This contrasts with the deterministic NF^2 model which is still value based.

For efficient processing of PRA expressions, there are two possible directions in query optimization. As stated before, all equivalences from PRA also hold in RA. Thus, it is possible to apply the same optimization strategies as in ordinary relational database management systems (for a

survey, see Jarke and Koch [1984]). Since most of these methods focus on minimizing the number of I/O operations required for computing the result, the assumptions underlying these strategies also hold in the case of a probabilistic relational database. A different approach is based on the observation that users of IR systems hardly ever want to see a ranked output list as a whole. Mostly, they are interested in a few top-ranking documents. In PRA, this means that the user wants to see only those tuples from the result relation which have the highest probabilities. Thus, it may be inefficient to compute the whole result relation first and then pick those tuples the user is interested in. A more efficient strategy is based on the idea of focusing on the result tuples with high probabilities and doing the processing only for these tuples.[5] This strategy can be supported by access structures which retrieve tuples in the order of decreasing probabilities. For example, inverted lists for index terms can be sorted this way; for vague predicates on numeric attributes, a B-tree structure can deliver the records according to decreasing probabilities. This strategy is described and evaluated in Pfeifer and Fuhr [1995].

Specific access structures also should be used for the implementation of vague predicates. For example, phrase search can be implemented by using inverted lists (for single words) with additional position information. Phonetic search can be sped up by using a phonetic index (i.e., inverted lists for phonetic codes).

Besides I/O complexity, the CPU time required for computing the tuple probabilities also should be considered in query optimization. As mentioned before, it is possible to determine for a given PRA expression whether or not the probabilities can be computed correctly by means of simple evaluation. Thus, a minimum extension of current relational DBMS technology could implement only simple evaluation; if there is no PRA expression for an SQL query for which simple evaluation works, the system could refuse to process such a query (or give a warning to the user). Similar considerations also hold for the handling of vague predicates. In a typical application where the only form of imprecision supported is probabilistic document indexing, there will be hardly any user query interfering with this restriction.

In a system that is able to handle event expressions, processing of event expressions should be restricted to PRA expressions where simple evaluation fails. Even in these cases, it is often possible to handle subexpressions by means of simple evaluation; thus, the full computational complexity of the inclusion-exclusion formula is hardly ever encountered in processing of PRA expressions.

For the problem of dependent events stemming from vague predicates, additional dependence information will hardly ever be available. However, a system based on event expressions will be able to identify tuples with dependent events in their event expression and flag them as erroneous.

[5]Of course, a certain number of other tuples also have to be considered, in order to ensure that the tuples presented as the result are the top-ranking ones.

Since vague predicates only aim at modeling a user's intentions, it is finally up to the user to decide about the quality of tuples marked as erroneous.

Comparing these efficiency issues with those of current experimental IR systems, a reader may come to the conclusion that our approach based on PRA represents a rather inefficient method for processing typical IR queries. However, it is fairly easy to detect PRA expressions representing these typical queries and to process them with the same strategies as in a pure IR system—even with the same access structures. So, the goal of our approach is the combination of efficiency of IR systems with the expressiveness of relational algebra.

As a standard interface to a database system implementing PRA, SQL seems to be feasible. This offers the possibility of easy integration of IR systems with other applications (e.g., word processors) in the same way as today's DBMS do. On the other hand, using an SQL interface does not mean that end-users will have to formulate SQL queries. It is possible to implement different user interfaces which allow for easier query formulation. Then user queries have to be translated into the relational query language offered by the underlying system. Vice versa, the relation computed as the answer to the query may be presented to the user in a different form (e.g., tuples representing documents should be displayed in their typical document format). For example, the approach presented in Desai et al. [1987] uses non-first-normal-form universal relations as the interface to an IR database: here the user views the database schema as one single (universal) relation, but attributes may be set valued. (This is very similar to commercial IR systems, where a document consists of a fixed number of fields, which in turn are set valued.) Queries are formulated with respect to the universal relation, and then they are translated into queries with the underlying database scheme. Query formulation can be simplified further by using a forms-based interface where the user types his or her conditions for the different attributes into the corresponding fields (e.g., see Pfeifer et al. [1995]).

Currently, we are implementing two variants of the basic approach presented in this article. In order to support an even more expressive query language than PRA (or SQL), we are working on the implementation of a probabilistic version of Datalog, which also allows for recursive queries (see Fuhr [1995]). Here Datalog queries are transformed into sets of PRA expressions which are processed iteratively. The second system under development is based on the probabilistic NF^2 model mentioned earlier in this article. First prototypes of both systems (although without any optimization yet) are already available.

8. RELATED WORK

8.1 Current IR systems

In comparison to the majority of today's (experimental as well as commercial) IR systems, PRA is able to express almost any function of these

systems. As we have shown, Boolean queries (in combination with probabilistic indexing) as well as probabilistic query term weighting and even phrase search can be formulated in PRA. Furthermore, PRA also can support retrieval in hypertext structures. Assume that we have a relation Link (Source, Dest) giving links from Source to Dest. Now we may assume that a document also is about a term if it has a link pointing to another document indexed with this term. Then a query for documents about IR can be expressed by forming the union of documents with the index term 'IR' and those having a link to another document indexed with 'IR':

$$\prod_{\text{DocNo}}(\sigma_{\text{Term}='\text{IR}'}(\text{DocTerm})) \cup \prod_{\text{Source}\rightarrow\text{DocNo}}(\sigma_{\text{DocNo}=\text{Dest}}(\sigma_{\text{Term}='\text{IR}'}(\text{DocTerm}) \bowtie \text{Link})).$$

(Here the projection parameter "Source → DocNo" denotes renaming of attribute Source to DocNo.) With this functionality, PRA offers a similar expressiveness as for example the INQUERY system (see Haines and Croft [1993]). Like this system, PRA implements retrieval as uncertain inference by computing the probability $P(d \rightarrow q)$ that document d implies query q. On the other hand, PRA does not deal with the problem how the underlying probabilities (of the basic events) are estimated, since this is outside of the scope of the model. However, there are other approaches which can perform this task, e.g., probabilistic text indexing presented in Fuhr and Buckley [1991] or the probabilistic weighting of vague predicates proposed in Fuhr [1990].

The major improvement of PRA over today's IR systems is its significantly higher expressiveness. As in RA, PRA allows a user to ask for any item stored in the database (see the examples shown in Section 3.2). Most experimental IR systems only support searching for documents. Some commercial IR systems offer functions for browsing the dictionary or extracting survey information (e.g., term statistics) from a set of documents that was retrieved in response to another query.[6] If a user seeks other types of objects, these systems offer very little help. For example, the query about authors writing about both IR and DB cannot be processed by any of these systems. As another example, the CORDIS database of the European commission contains information about research groups in Europe, their members, and their publications. This situation can be modeled by adding a relation Affil (Institute, AName) to the relations DocTerm (DocNo, Term) and DocAu (DocNo, AName) from Section 3.1. Then we can ask for institutes doing research both in IR and DB by means of the expression

$$\prod_{\text{Institute}}(\text{Affil} \bowtie \text{DocAu} \bowtie \sigma_{\text{Term}='\text{IR}'}(\text{DocTerm})) \cap \prod_{\text{Institute}}(\text{Affil} \bowtie \text{DocAu} \bowtie \sigma_{\text{Term}='\text{DB}'}(\text{DocTerm})).$$

[6]However, due to a weak theoretical basis of the query language, these additional features are introduced as additional query language constructs, thus leading to a rather complex query language (which makes it unsuitable for many potential users).

Survey information for sets of documents (e.g., the sets of terms or authors occurring in these documents) can be derived in PRA by means of the projection operator. (However, it would be useful to extend the basic algebra with some aggregate functions like SUM or COUNT in SQL—which would yield expectations as attribute values.)

Besides a higher expressiveness, PRA is also able to support data security by means of views. As a simple example, assume that we have a security classification of documents stored as a relation DocSec(DocNo,SecLev), where SecLev is an integer giving the security level for the document with the number DocNo. If certain users are allowed to view only documents with a security level ≤ 2, then two specific views can be defined for them by means of the expressions

$$\text{DocTerm2} = \prod_{\text{DocNo,Term}}(\sigma_{\text{SecLev}\leq 2}(\text{DocSec})) \bowtie \text{DocTerm}$$

and

$$\text{DocAu2} = \prod_{\text{DocNo,AName}}(\sigma_{\text{SecLev}\leq 2}(\text{DocSec})) \bowtie \text{DocAu}.$$

Now this user is allowed to issue queries with respect to the two relations DocTerm2 and DocAu2 only. Thus, he or she cannot get any information about the documents he or she is not allowed to see. Of course, no global dictionary can be supported in this setting; however, the set of terms occurring in all visible documents can be derived by means of the projection $\Pi_{\text{Term}}(\text{DocTerm2})$.

On the other hand, we do not claim that the relational model is good at modeling the internal structure of documents. A survey over text data models of this kind is given in Loeffen [1994].

8.2 Integration of IR and Database Systems

The integration of IR and database systems has been a research topic for several years now. A number of authors have discussed the application of standard relational algebra in IR systems (see Desai et al. [1987] and Macleod [1991]). These papers stress the capability to search for all kinds of objects in the database and the advantages of using a standard database approach, but the approaches lack any ranking facility. Blair [1988] presents a model for overcoming this weakness; weights are treated like ordinary attributes here, and ranking is achieved via the ORDER BY clause of SQL. However, the combination formulas for weights have to be stated explicitly in the query formulation. In contrast, our approach uses weights implicitly, and the combination formulas are determined by the relational operator. Furthermore, we assume that probabilistic relations are implicitly ranked according to descending probabilities, so no explicit ordering operator is necessary for most queries.

It has been recognized already in Schek and Pistor [1982] that non-first-normal-form (NF^2) relations are suited much better to IR problems than the standard relational model. Its major advantages are easier query

formulation for most problems and more efficient processing. On the other hand, there are only minor differences in terms of expressiveness. For this reason, we want to study the general problems with a PRA by using first-normal-form relations before we switch to the NF^2 model.

There are also some approaches for extending text retrieval methods in order to cope with facts, too. If we concentrate on approaches that apply ranking methods at least for texts, then three different methods for the combination of text and fact retrieval can be distinguished:

—*Boolean retrieval for facts*: Raghavan et al. [1986] describe the combination of probabilistic text retrieval methods with Boolean fact retrieval. Here the fact conditions select a set of objects from the database, followed by a ranking of objects according to the text conditions.

—*Non-Boolean retrieval for facts with binary weighting for fact conditions*: A simple approach for applying ranking methods to fact conditions as well as to text conditions is to treat fact conditions like index terms in combination with binary indexing. In the SMART system [Buckley 1985], different so-called "concept types" can be distinguished; besides text terms, for example, dates or names of persons or institutions can be used for retrieval. However, for these attributes, only tests on equality can be performed. In the approach presented by Saxton and Raghavan [1990], an (IR-type) query is a disjunction of conditions, where a single condition can be a Boolean combination of subconditions. For fact conditions, the standard predicates and comparison operators can be applied. Each condition in the query is given a weight, and then objects are ranked according to the weighted sum of the conditions they are fulfilling. Rabitti and Savino [1990] describe an advanced system for retrieval of multimedia data (including facts). Their system uses probabilistic retrieval for text as well as for facts. Both kinds of conditions can be assigned weights with respect to the query, and text terms also can be given probabilistic index terms weights with respect to an object.

—*Non-Boolean retrieval for facts with nonbinary weighting for fact conditions*: Fuhr [1992a] presents a model for combining probabilistic text retrieval, vague fact conditions, and imprecise data. However, this approach is based on a linear data model only.

To sum up the different efforts for the integration of IR and database systems developed in the past, it can be concluded that approaches based on standard data models do not pay attention to the intrinsic uncertainty and vagueness of IR; on the other hand, approaches aiming to extend IR models for coping with facts, too, offer only a very poor data model and/or a query language with limited expressiveness.

8.3 Imprecision in Databases

A number of approaches for coping with imprecision in databases have been developed in the past; for a survey, see Kim [1989] and Motro [1990]. A probabilistic relational model with extensional semantics is described by

Cavallo and Pittarelli [1987]. In this model, too, probabilistic weights are assigned to tuples, but tuples in the same relation always represent disjoint events. So this model can be regarded as a special case of our approach, and thus the application range is rather limited.

In a certain sense, the PDM model described by Barbara et al. [1992] can be regarded as a further development of the Cavallo and Pittarelli model, in order to overcome some of its disadvantages. This model is similar to a nested relational model, where tuples of inner relations represent imprecise attribute values. On the other hand, no weights can be assigned to outermost tuples, so this approach lacks a ranking facility. For queries with selection conditions relating to attributes with imprecise values, one can specify a minimum probability with which an atomic condition should be true. However, it is not possible to combine probabilities relating to different conditions. By restricting the set of possible operations, the PDM guarantees that only correct results can be produced; thus, many operations possible in PRA cannot be performed within the PDM. The representation of imprecise attribute values in PDM is more intuitive; however, PRA is more expressive by allowing also relations with independent tuples.

Whereas both of these models are based on Bayesian probability theory, the approach presented by Lee [1992] uses Dempster-Shafer theory. Here weights can be assigned to attribute values as well as to tuples as a whole. Imprecise attribute values are represented as sets of values with associated probabilities; thus, relations are not in first-normal form. However, since no real nesting of relations is allowed (as in the PDM model), it is not possible to represent imprecise attribute values which are dependent. Furthermore, since this model is based on extensional semantics, it is not a generalization of relational algebra; most equivalences of relational algebra do not hold within this model.

There is also a number of data models based on fuzzy theory (e.g., Prade and Testemale [1984], Takahashi [1993], and Lee and Kim [1993]). In principle, probabilistic and fuzzy theory approaches are orthogonal to each other, and the choice of the appropriate theory depends on the actual application. Furthermore, since fuzzy theory is based on extensional semantics, equivalences from relational algebra do not hold for fuzzy relational algebra in general.

The problem of imprecise attribute values in the form of null values or disjunctive information has been discussed extensively in the database literature (e.g., see Lipski [1979], Vassiliou [1979], Reiter [1984], and Imielinski and Lipski [1984]); Imielinski [1989] gives a brief survey over this work. As all these approaches are based on two-valued logic, the correct treatment of imprecise values is obvious (e.g., in the proof-theoretic approach [Reiter 1984], it follows from the axioms of first-order logic). In Codd [1986], a three-valued logic is used instead, in order to retrieve "maybe" answers in addition to the correct answers of a query. This model can be regarded as a very simple ranking mechanism. Modifications of this approach are discussed in Gessert [1990] and Yue [1991].

9. CONCLUSIONS

In this article, we have described a probabilistic relational algebra which is a generalization of standard relational algebra. It allows modeling of different kinds of imprecision in databases. Following the concept of intensional semantics, our approach keeps track of the basic events that contribute to a certain tuple of a relation, and thus all probabilities are computed correctly. In comparison to other approaches with extensional semantics, there is no loss in efficiency for relational expressions where both approaches yield the same result, and there is only an overhead introduced by means of the event expressions when extensional semantics yields incorrect results (according to probability theory).

In contrast to other probabilistic data models, PRA is a generalization of relational algebra. Thus, descriptive query languages based on relational calculus can be used as an interface to a system based on PRA. Furthermore, the algebra offers the possibility of query optimizations.

From a theoretical point of view, the independence assumptions used throughout this article may look rather restrictive. Of course, it is no problem to apply PRA in conjunction with more general assumptions about the dependence of events. However, we think that our choice of independence assumptions is realistic for a broad range of applications. On the other hand, the limitations of our approach (especially for vague predicates) are not a theoretical construct only; they relate directly to practical problems in estimating the stochastical dependence information.

PRA allows a close integration of IR and database systems, since it includes basic IR methods such as probabilistic document indexing and search term weighting. Through the combination of textual and factual data, PRA supports queries which are not possible in current IR or database systems. For factual data alone, there are powerful mechanisms for modeling imprecise data and vague queries.

It should be emphasized that PRA represents a logical data model. Thus, it makes no assumptions about the underlying physical data model. For example, in an integrated IR and database system, it may be appropriate to store textual and factual data in a different way along with different access paths. The strength of our approach, however, is that it offers the user a unified view of the integrated system, namely a generalization of the relational model.

Finally, using the relational model as a standard interface to an integrated IR and database system does not mean that end-users will have to use SQL to query such a system. Rather, such a standard interface offers the possibility of easy implementation of different user-friendly interfaces on top of this system interface.

APPENDIX

PROOF OF THEOREM 3.3.1. In Section 3.3, we have defined a PRA relation t and an RA relation $\bar{t}$ with identical schemas to be equivalent iff $\bar{t} \cong t \Leftrightarrow \forall \mu \, (\mu \in \bar{t} \Leftrightarrow t(\mu) = \top)$.

Now we show that for the five basic operations of RA that PRA always yields equivalent results. Instead of the natural join, we only consider the cartesian product here, since any join can be expressed as a combination of cartesian product and selection.

In the following we assume $\bar{r} \cong r$ and $\bar{s} \cong s$.

Union

$$\bar{t} = \bar{r} \cup \bar{s} \wedge t = r \cup s \Rightarrow \bar{t} \cong t$$

In RA, $\mu \in \bar{r} \cup \bar{s}$ is defined as $\mu \in \bar{r} \vee \mu \in \bar{s}$. Applying the definition of equivalence for ordinary and probabilistic relations gives $r(\mu) = \top \vee s(\mu) = \top$. Since the event expressions form a Boolean algebra, we get $t(\mu) = r(\mu) \veebar s(\mu) = \top$, if $\mu \in \bar{r} \cup \bar{s}$.

In PRA, $(r \cup s)(\mu)$ is defined as $r(\mu) \veebar s(\mu)$. Since the event expressions form a Boolean algebra and since $r(\mu) \in \{\top, \bot\}$ and $s(\mu) \in \{\top, \bot\}$ hold, we get $r(\mu) \veebar s(\mu) = \top$ for all tuples $\mu \in \bar{r} \cup \bar{s}$ and $r(\mu) \veebar s(\mu) = \bot$ for all tuples $\mu \notin \bar{r} \cup \bar{s}$.

Thus $\bar{t} \cong t$ holds.

Difference

$$\bar{t} = \bar{r} - \bar{s} \wedge t = r - s \Rightarrow \bar{t} \cong t$$

In RA, $\mu \in \bar{r} - \bar{s}$ is defined as $\mu \notin \bar{r} \wedge \neg\mu \in \bar{s}$. Applying the definition of equivalence for ordinary and probabilistic relations gives $r(\mu) = \top \wedge s(\mu) = \bot$. Since the event expressions form a Boolean algebra, we get $t(\mu) = r(\mu) \barwedge \neg s(\mu) = \top$, if $\mu \in \bar{r} - \bar{s}$.

In PRA, $(r - s)(\mu)$ is defined as $r(\mu) \barwedge \neg s(\mu)$. Since the event expressions form a Boolean algebra and since $r(\mu) \in \{\top, \bot\}$ and $s(\mu) \in \{\top, \bot\}$ hold, we get $r(\mu) \barwedge \neg s(\mu) = \top$ for all tuples $\mu \in \bar{r} - \bar{s}$ and $r(\mu) \barwedge \neg s(\mu) = \bot$ for all tuples $\mu \notin \bar{r} - \bar{s}$.

Thus $\bar{t} \cong t$ holds.

Cartesian Product

$$\bar{t} = \bar{r} \times \bar{s} \wedge t = r \times s \Rightarrow \bar{t} \cong t$$

In RA, $\mu \in \bar{r} \times \bar{s}$ is defined as $\mu[A] \in \bar{r} \wedge \mu[B] \in \bar{s}$. Applying the definition of equivalence for ordinary and probabilistic relations gives $r(\mu[A]) = \top \wedge s(\mu[B]) = \top$. Since the event expressions form a Boolean algebra, we get $t(\mu) = r(\mu[A]) \barwedge s(\mu[B]) = \top$, if $\mu \in \bar{r} \times \bar{s}$.

In PRA, $(r \times s)(\mu)$ is defined as $r(\mu) \barwedge s(\mu)$. Since the event expressions form a Boolean algebra and since $r(\mu) \in \{\top, \bot\}$ and $s(\mu) \in \{\top, \bot\}$ hold, we get $r(\mu) \barwedge s(\mu) = \top$ for all tuples $\mu \in \bar{r} \times \bar{s}$ and $r(\mu) \barwedge s(\mu) = \bot$ for all tuples $\mu \notin \bar{r} \times \bar{s}$.

Thus $\bar{t} \cong t$ holds.

Selection

$$\bar{t} = \sigma_{X\Theta x}(\bar{r}) \wedge t = \sigma_{X\Theta x}(r) \Rightarrow \bar{t} \cong t$$

In RA, $\mu \in \sigma_{X\Theta x}(\bar{r})$ is defined as $\mu \in \bar{r} \wedge \mu(X)\Theta x$. Applying the definition of equivalence for ordinary and probabilistic relations gives $r(\mu) = \top \wedge \mu(X)\Theta x$. We get $t(\mu) = r(\mu) = \top$, if $\mu \in \sigma_{X\Theta x}(\bar{r})$.

In PRA, $\sigma_{X\Theta x}(r)$ is defined as $r(\mu)$, if $\mu(X)\Theta x$ is true and $\perp$ otherwise. Since $r(\mu) \in \{\top, \perp\}$, we get $t(\mu) = r(\mu) = \top$ for all tuples $\mu \in \sigma_{X\Theta x}(\bar{r})$ and $t(\mu) = \perp$ for all tuples $\mu \notin \sigma_{X\Theta x}(\bar{r})$.

Thus $\bar{t} \cong t$ holds.

Projection

$$\bar{t} = \prod_Y(\bar{r}) \wedge t = \prod_Y(r) \Rightarrow \bar{t} \cong t$$

In RA, $\mu \in \Pi_Y(\bar{r})$ is defined as $\exists v \in \bar{r}(v = \mu\,[Y])$. Applying the definition of equivalence for ordinary and probabilistic relations gives $\exists v(r(v) = \top \wedge v = \mu[Y])$. We get $t(\mu) = \veebar_{v \in Tup(A) \wedge v = \mu[Y]}\, r(v) = \top$, if $\mu \in \Pi_Y(\bar{r})$.

In PRA, $\Pi_Y(r)(\mu)$ is defined as $\veebar_{v \in Tup(A) \wedge v = \mu[Y]}\, r(v)$. Since $r(v) \in \{\top, \perp\}$, we get $t(\mu) = \top$, if $\exists v(r(v) = \top \wedge v = \mu\,[Y])$, i.e., if $\mu \in \Pi_Y(\bar{r})$. Otherwise we get $t(\mu) = \perp$.

Thus $\bar{t} \cong t$ holds. □

REFERENCES

BARBARA, D., GARCIA-MOLLINA, H., AND PORTER, D. 1992. The management of probabilistic data. *IEEE. Trans. Knowl. Data Eng. 4*, 5, 487–502.

BILLINGSLEY, P. 1979. *Probability and Measure*. John Wiley and Sons, New York.

BLAIR, D. C. 1988. An extended relational document retrieval model. *Inf. Process. Manage. 24*, 3, 349–371.

BUCKLEY, C. 1985. Implementation of the SMART information retrieval system. Tech. Rep. 85-686, Dept. of Computer Science, Cornell. Univ., Ithaca, N.Y.

CAVALLO, R. AND PITTARELLI, M. 1987. The theory of probabilistic databases. In *Proceedings of the 13th International Conference on Very Large Databases*. Morgan Kaufmann, San Mateo, Calif., 71–81.

CODD, E. F. 1986. Missing information (applicable and inapplicable) in relational databases. *SIGMOD Rec. 15*, 4, 53–78.

CROFT, W., TURTLE, H., AND LEWIS, D. 1991. The use of phrases and structured queries in information retrieval. In *Proceedings of the 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 32–45.

DESAI, B., GOYAL, P., AND SADRI, F. 1987. Non-first-normal form universal relations: An application to information retrieval systems. *Inf. Syst. 12*, 1, 49–55.

FLICKNER, M., SAWHNEY, H., NIBLACK, W., ASHLEY, J., HUANG, Q., DOM, N., GORKANI, M., HAFNER, J., LEE, D., PETKOVIC, D., STEELE, D., AND YANKER, P. 1995. Query by image and video content: The QBIC system. *Computer 28*, 9, 23–32.

FUHR, N. 1990. A probabilistic framework for vague queries and imprecise information in databases. In *Proceedings of the 16th International Conference on Very Large Databases*, D. McLeod, R. Sacks-Davis, and H. Schek, Eds. Morgan Kaufmann, San Mateo, Calif., 696–707.

FUHR, N. 1992a. Integration of probabilistic fact and text retrieval. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Infor-*

mation Retrieval, N. Belkin, P. Ingwersen, and M. Petjersen, Eds. ACM, New York, 211–222.

FUHR, N. 1992b. Probabilistic models in information retrieval. *Comput. J. 35*, 3, 243–255.

FUHR, N. 1993. A probabilistic relational model for the integration of IR and databases. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 309–317.

FUHR, N. 1995. Probabilistic Datalog—A logic for powerful retrieval method. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 282–290.

FUHR, N. AND BUCKLEY, C. 1991. A probabilistic learning approach for document indexing. *ACM Trans. Inf. Syst. 9*, 3 (July), 223–248.

FUHR, N. AND RÖLLEKE, T. 1996. A probabilistic NF2 relational algebra for integrated information retrieval and database systems. In *Proceedings of the 2nd World Conference on Integrated Design and Process Technology (IDPT)*. Soc. for Design and Process Science, Austin, Tex.

GESSERT, G. 1990. Four valued logic for relational database systems. *SIGMOD Rec. 19*, 1, 29–35.

GU, J., THIEL, U., AND ZHAO, J. 1993. Efficient retrieval of complex objects: Query processing in a hybrid DB and IR system. In *Proceedings 1. GI-Fachtagung Information Retrieval*. Universitätsverlag Konstanz, Konstanz, Germany.

HAINES, D. AND CROFT, B. 1993. Relevance feedback and inference networks. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, R. Korfhage, E. Rasmussen, and P. Willett, Eds. ACM, New York, 2–11.

HARMAN, D. 1995. Overview of the second text retrieval conference (TREC-2). *Inf. Process. Manage. 31*, 103, 271–290.

IMIELINSKI, T. 1989. Incomplete information in logical databases. *Data Eng. Bull. 12*, 2, 29–40.

IMIELINSKI, T. AND LIPSKI, W. 1984. Incomplete information in relational databases. *J. ACM 31*, 4, 761–791.

JARKE, M. AND KOCH, J. 1984. Query optimization in database systems. *ACM Comput. Surv. 16*, 111–152.

KIM, W., Ed. 1989. Special issue on imprecision in databases. *Data Eng. Bull. 12*, 2.

LEE, D. AND KIM, M. 1993. Accommodating subjective vagueness through a fuzzy extension to the relational data model. *Inf. Syst. 18*, 6, 363–374.

LEE, S. 1992. An extended relational database model for uncertain and imprecise information. In *Proceedings of the 18th VLDB Conference*. Morgan Kaufmann, San Mateo, Calif., 211–220.

LIPSKI, W. 1979. On semantic issues connected with incomplete information databases. *ACM Trans. Database Syst. 4*, 3, 262–296.

LOEFFEN, A. 1994. Text databases: A survey of text models and systems. *SIGMOD Rec. 23*, 1, 97–106.

MACLEOD, I. 1991. Text retrieval and the relational model. *J. Am. Soc. Inf. Sci. 42*, 3, 155–165.

MOTRO, A. 1988. VAGUE: A user interface to relational databases that permits vague queries. *ACM Trans. Off. Inf. Syst. 6*, 3, 187–214.

MOTRO, A. 1990. Accommodating imprecision in database systems: Issues and solutions. *SIGMOD Rec. 19*, 4, 69.

PEARL, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, Calif.

PFEIFER, U. AND FUHR, N. 1995. Efficient processing of vague queries using a data stream approach. In *Proceedings of the 18th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 189–198.

PFEIFER, U., FUHR, N., AND HUYNH, T. 1995. Searching structured documents with the enhanced retrieval functionality of freeWAIS-sf and Sfgate. In *Computer Networks and*

ISDN Systems: Proceedings of the 3rd International World-Wide Web Conference, D. Kroemker, Ed. Elsevier, Amsterdam, 1027–1036.

PRADE, H. AND TESTEMALE, C. 1984. Generalizing database relational algebra for the treatment of incomplete/uncertain information and vague queries. *Inf. Sci. 34*, 115–143.

RABITTI, F. AND SAVINO, P. 1990. Retrieval of multimedia documents by imprecise query specification. In *Advances in Database Technology—EDBT '90*, F. Bancilhon, C. Thanos, and D. Tsichritzis, Eds. Springer-Verlag, Berlin, 203–218.

RAGHAVAN, V., SAXTON, L., WONG, S., AND TING, S. 1986. A unified architecture for the integration of data base management and information retrieval systems. In *Information Processing 86*, H.-J. Kugler, Ed. Elsevier, Amsterdam, 1049–1054.

REITER, R. 1984. Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling*, M. Brodie, J. Mylopoulous, and J. Schmidt, Eds. Springer, New York.

RÖLLEKE, T. 1994. Equivalences of the probabilistic relational algebra. Tech. Rep., Dept. of Computer Science, Univ. of Dortmund, Dortmund, Germany.

SALTON, G. AND BUCKLEY, C. 1988. Term weighting approaches in automatic text retrieval. *Inf. Process. Manage. 24*, 5, 513–523.

SAXTON, L. AND RAGHAVAN, V. 1990. Design of an integrated information retrieval database management system. *IEEE Trans. Knowl. Data Eng. 2*, 2, 210–219.

SCHEK, H.-J. AND PISTOR, P. 1982. Data structures for an integrated database management and information retrieval system. In *Proceedings of the 8th International Conference on Very Large Data Bases*. Morgan Kaufmann, San Mateo, Calif., 197–207.

TAKAHASHI, Y. 1993. Fuzzy database query languages and their relational completeness theorem. *IEEE Trans. Knowl. Data Eng. 5*, 1, 122.

VAN RIJSBERGEN, C. J. 1986. A non-classical logic for information retrieval. *Comput. J. 29*, 6, 481–485.

VASSILIOU, Y. 1979. Null values in database management—A denotational semantics approach. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*. ACM, New York.

VOSSEN, G. 1991. *Data Models, Database Languages and Database Management Systems*. Addison-Wesley, Reading, Mass.

YUE, K. 1991. A more general model for handling missing information using a 3-valued logic. *SIGMOD Rec. 20*, 3, 43–49.

ZEMANKOVA, M. AND KANDEL, A. 1985. Implementing imprecision in information systems. *Inf. Sci. 37*, 107–141.

Received September 1994; accepted May 1996