The Third AFSS(1998), 128−133

Implementation issues for Uncertain Relational Databases

Hairong Yu, Arthur Ramer

School of Computer Science and Engineering, University of New South Wales, Sydney 2052, Australia
Tel:+61-2-9385-{3980,3978} Fax:+61-2-9385-1814 E-mail:{hairong,ramer}@cse.unsw.edu.au

Abstract: *This paper aims to present some ideas for implementation of Uncertain Relational Databases (URD) which are extensions of classical relational databases. Our system firstly is based on possibility distribution and probability theory to represent and manipulate fuzzy and probabilistic information, secondly adopts flexible mechanisms that allow the management of uncertain data through the resources provided by both available relational database management systems and front-end interfaces, and lastly chooses dynamic SQL to enhance versatility and adjustability of systems.*

Keywords: Relational databases, Possibility distribution, Relational algebra, First-order logic of probability.

1 Introduction

In the real world, information is not always precise and imperfect, and sometimes, might be missing or hard to get. When all these uncertain data involve, many researchers have been desirable to illustrate it in the databases so that it could be used to answer queries of interest as much as possible for many years [7, 25, 1, 5, 21, 20, 2, 15, 9, 11, 30].

We propose a generalised model: an uncertain relational database (URD), basically non-first normal form, to handle uncertain information in relational databases. As all kinds of uncertainty: imprecision, fuzziness, incompleteness, vagueness, inconsistency, ambiguity, etc could be classified into two group: possibilistic and probabilistic. URD is made to use of possibility distributions attached with attributes and probabilities affiliated to tuples to depict and compute those various uncertainness.

A prototype called Uncertain Relational Database System (URDS) is an experimental model which puts URD into practice. The objective to build URDS is that it must be capable of retrieving information stored in conventional database systems plus uncertain data processing. Owning to the fact that a major number of database application system has been developed using relational database system and written in SQL which is a widely accepted and international standard. These resources cannot be neglected in proposing URDS.

A system which is based on classical relational databases and its declarative query language with additional requirement to be able to process uncertain values of attributes and predicates, is outlined initially.

The system is written by C as the host language and embedded SQL to deal with a traditional database engine which is Oracle V7.3 under IRIX on SGI workstation.

The rest of paper is organised as follows. In next section, we give a brief review of URD model, and express the approach to URD algebra. Section 3 is devoted to overview URDS architecture and its components. In section 4, more detail for the system is showed. And section 5 contains some concluding remarks.

2 URD Model

The URD model describes how real-world data can be conceptually represented as computerised information. It also includes the type of operations available to access and update the information.

The constituent parts of a URD are a set of uncertain relations comprised of tuples, similarly to an ordinary relational database. However each tuple, or its fields, may have a richer structure than standard relational tuples.

We define uncertain relation as generalisation of Codd's relation model of data [24] by associating possibility with attributes to express possibilistic data and an additional value $p(\mathbf{t})$ to indicate the probability that tuple $\mathbf{t}$ belongs to an uncertain relation $\mathcal{R}$.

Definition 1
An uncertain database $\mathcal{D}$ is defined as a set of uncertain relations $\mathcal{R}_i$, where $i = 1, 2, \ldots, n$, i.e. $\mathcal{D} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n\}$.

Definition 2
Let $R(A_1, A_2, \ldots, A_m)$ be a relation schema where $\text{dom}(A_i)$ is the domain of A_i, for $i = 1, 2, \ldots, m$. An *uncertain relation* $\mathcal{R}$ of relation R is defined as follows:

$$\mathcal{R} \subseteq \{(\mathbf{t}, p(\mathbf{t})) \mid \mathbf{t} = (a_1, a_2, \ldots, a_m) \in DOM(A_1) \times DOM(A_2) \times \cdots \times DOM(A_m), p(\mathbf{t})\}$$

Where $p(\mathbf{t}) \in [0,1]$ and symbol $\subseteq$, $\times$ and $\in$ denote the subset, Cartesian product and member respectively in ordinary set theory. And $DOM(A_i) = \Pi(\mathcal{U}_i) \cup void$. $\Pi(\mathcal{U}_i)$, $i = 1, 2, \ldots, m$, are collections of all possibility distributions on a universe of discourse $\mathcal{U}_i$, here *void* is a special value to be interpreted as not yet defined, meaning that when we know more we intend to fill in the value. It is assumed that key attributes take ordinary nonfuzzy values and all uncertain relations are finite. We would also like to have special symbols to denote concepts like 'unknown' or 'unavailable', *none* interpreted as no possibility for the value of an object could exist, and *unknown* a possible possibility for any value of an object.

Recall from [28], the possibility distribution function $\pi_{A(x)}$ associated with an attribute A of an object x is defined as:

$$\pi_{A(x)} : \mathcal{U} \to [0,1]$$

where $\mathcal{U}$ is a universe of discourse of $A(x)$ and $\pi_{A(x)}(u)$ represents the possibility that $A(x)$ supposes value u in $\mathcal{U}$. We use

$$\pi_{A(x)}(\mathcal{U}) = \{\pi_{A(x)}(u_1)/u_1, \pi_{A(x)}(u_2)/u_2, \ldots, \pi_{A(x)}(u_j)/u_j\}, \text{ where } u_j \in \mathcal{U}, j = 1, 2, \ldots, n.$$

A tuple $(\mathbf{t}, p(\mathbf{t}))$ in an uncertain relation $\mathcal{R}$ means a probability $p(\mathbf{t})$ is associated with fact $\mathbf{t}$. Conversely, the probability of $\mathbf{t}$ not belonging to $\mathcal{R}$ is $1 - p(\mathbf{t})$.

URD stores uncertain data directly within databases in order to handle the information more efficiently than the system which poses uncertain queries based on crisp data.

This paper largely explains how URD can be realized. The complete aspects of theoretical background are stated in [22, 26, 27, 23].

Uncertain Relational Algebra

The relational algebra operations are usually divided into two groups [6, 10]: (a) mathematical set theory operations: Union, Intersection, Difference and Cartesian Product. (b) operations developed specifically for relational databases: Select, Project and Join. We must broaden them to uncertain relations.

Consider two distinctive n-ary uncertain relations $\mathcal{R}$ and $\mathcal{S}$ in a universe of discourse $\mathcal{U} = \{\mathcal{U}_1 \times \mathcal{U}_2 \times \cdots \times \mathcal{U}_n, p\}$.

Union

$$\mathcal{R} \cup \mathcal{S} = \{max(\pi_{\mathcal{R}}(u_i), \pi_{\mathcal{S}}(u_i))/u_i, (p_{\mathcal{R}} + p_{\mathcal{S}} - p_{\mathcal{R}} \times p_{\mathcal{S}}) : u_i \in \mathcal{U}_i\}, \text{ where } i = 1, 2, \ldots, n.$$

Intersection

$$\mathcal{R} \cap \mathcal{S} = \{min(\pi_{\mathcal{R}}(u_i), \pi_{\mathcal{S}}(u_i))/u_i, p_{\mathcal{R}} \times p_{\mathcal{S}} : u_i \in \mathcal{U}_i\} \text{ where } i = 1, 2, \ldots, n.$$

Difference

$$\mathcal{R} - \mathcal{S} = \{max((\pi_{\mathcal{R}}(u_i) - \pi_{\mathcal{S}}(u_i)), 0)/u_i, p_{\mathcal{R}}(1 - p_{\mathcal{S}}) : u_i \in \mathcal{U}_i\}, \text{ where } i = 1, 2, \ldots, n \text{ and } - \text{ denotes ordinary subtraction.}$$

Cartesian product

$$\mathcal{R} \times \mathcal{S} = \{\mathbf{t} = (a_1, \ldots, a_m, b_1, \ldots, b_n), p_{\mathcal{R}} \times p_{\mathcal{S}} : a_i \in \mathcal{U}_i, b_j \in \mathcal{V}_j, 0 \le i \le m, 0 \le j \le n\}, \text{ where } \mathcal{R} \in \mathcal{U}_1 \times \mathcal{U}_2 \times \cdots \times \mathcal{U}_m \text{ and } \mathcal{S} \in \mathcal{V}_1 \times \mathcal{V}_2 \times \cdots \times \mathcal{V}_n.$$

Selection

$$\sigma_{C(A_{i_1}, A_{i_2}, \ldots, A_{i_k}, \theta)}(\mathcal{R}) = \{\mathbf{t}, p(\mathbf{t}) \mid sup[\pi_{\mathcal{U}_{i_j}}(u)\mu_{A_{i_j}}(u)] \ge \theta : u \in \mathcal{U}, p(\mathbf{t}) \in \mathcal{R}, 1 \ge \theta \ge 0, 1 \le j \le k\},$$

where μ is membership function of fuzzy set A_{i_j}, θ threshold value and $\mathcal{C}$ k-ary predicate from $\mathcal{U}_{i_1}, \mathcal{U}_{i_2}, \ldots, \mathcal{U}_{i_k}$.

Projection

Let $\mathcal{R}$ have attributes $A_1, A_2, \ldots, A_m$, the new relation $\mathcal{W}$ is projection of $\mathcal{R}$ only onto attributes $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$, where $k < m$, is defined as:

$$\pi_{A_{i_1}, A_{i_2}, \ldots, A_{i_k}}(\mathcal{R}) = \{\mathcal{W}(A_{i_1}, A_{i_2}, \ldots, A_{i_k}, p_{\mathcal{W}}(\mathbf{t})) : p_{\mathcal{W}}(\mathbf{t}) = p_{\mathcal{R}}(\mathbf{t}), \{A_{i_1}, \ldots, A_{i_k}\} \subseteq \{A_1, \ldots, A_m\}\}.$$

Natural Join

Let m-ary $\mathcal{R}$ and n-ary $\mathcal{S}$ join over attributes $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$ in $\mathcal{R}$ and $B_{i_1}, B_{i_2}, \ldots, B_{i_k}$ in $\mathcal{S}$ and one threshold value θ, the natural join operation which produces $(m + n - k)$-ary relation is defined as:

$$\mathcal{R} \bowtie_{A_{i_1}, \ldots, A_{i_k}, B_{i_1}, \ldots, B_{i_k}} \mathcal{S} = \{\mathbf{t} = (a_1, \ldots, a_m, b_1, \ldots, b_n) - (a_{i_1}, \ldots, a_{i_k}), p_{\mathcal{R}}(\mathbf{t}) \times p_{\mathcal{S}}(\mathbf{t}) : sup[min(\pi_{A_{i_l}}(u), \pi_{B_{i_l}}(u))] \ge \theta,$$

built interactively with input from users wanting to have little or no knowledge about perfect attribute values of SQL.

4 More aspects of URDS

Datatype

Besides numerical type, the form of data represented uncertain conception which is allowed to store as column values in Oracle tables is only character string internal datatype. Others are for number, date or tag. As a result, VARCHAR2, variable-length character string, that length is less than 64K bytes, is most suitable datatype. Tables below are the description and sample contents of relation students, where the key attribute S_ID which must not an uncertain attribute specifies a constraint definition. This effect is that only data which satisfies the constraint condition is stored to the table.

```
SQL> desc students
 Name              Null?         Type
 ----------------  ------------  ---------------
 S_ID              NOT NULL      NUMBER(4)
 S_NAME                          VARCHAR2(10)
 AGE                             VARCHAR2(20)
 COURSE                          VARCHAR2(20)
 PROBABILITY                     NUMBER(6,4)
```

```
SQL> select * from students;
... Please enter RETURN to continue ...

 S_ID  S_NA     AGE        COURSE           PR
 ----  ------   --------   --------------   --
 2104  Daniel   about 30   none             .9
 2166  Anna     29         PhD              .2
 2289  Stefan   void       PhD or Master    1
 1677  Karen    20 or 21   void             .8
 2347  John     unknown    Undergraduate    .5
```

Here S_NA stands for S_NAME and PR for attribute PROBABILITY attribute names.

All the above fuzzy attributes: about 30, 20 or 21 and unknown etc are defined by possibilistic distribution and saved at data library as $\{\frac{0.8}{29}, \frac{1}{30}, \frac{0.9}{31}\}$, $\{\frac{1}{20}, \frac{1}{21}\}$ and $\pi_{Age(John)}(u) = 1$ where $u \in \mathcal{U}_{Age}$.

Data Library

Data library consists of all the specification of data written in structure, array and pointer type of C language:

- finite attribute domains,
- every special value of designated elements in each domain,
- fuzzy membership functions.

For competent practice, the ad hoc clarification for individual application or user preferences could be added directly into the library at any time.

The ability to dynamically scale the possibilistic distribution enables meaningful uncertain information processing for a much larger set of applications and wider set of users, and reduces the amount of database specific knowledge required from users.

Function Library

Function library mainly conduct:

- calculation of possibilistic distribution or probability values,
- predicates computing the grade value between two uncertain data,
- extended connectives between two uncertain formulas, such as $\wedge_\tau, \vee_\tau$ *and* $\neg_\tau$ mean that the predicate is true to the degree of τ,
- extended comparators which is an extension from the arithmetic comparison operation (eg. $=, \neq, \leq, <, \geq, >$),
- mathematical aggregate function on collections of values from the database (eg. SUM, AVERAGE, MAXIMUM, MINIMUM and COUNT),
- linguistic modifiers like very, more or less, quite and of course uncertain relational algebra which is mentioned at section 2.

All of the above is maintained as some archive files written by C to form runtime function library.

Dynamic SQL

The dynamic SQL facility is designed exclusively for supporting online applications. Our application could be characterised by a great deal of variability, instead of writing a specific SQL query for every possible condition, we find it much more convenient to construct parts of the SQL query dynamically at running time and then to bind and execute them dynamically.

For example, the system just simply prompt users for a search condition to be used in the WHERE clause of a SELECT or DELETE statement. And users could choose from menus listing SQL operations, table names, column names, and so on. Therefore it is an extremely flexible system.

With all methods, dynamic SQL statements must be stored in a character string, which must be a host variable [1], that is the key to communication between host program and Oracle, or quoted literal.

[1]Oracle stores input data in database columns, and stores output data in program host variables.

$1 \leq i \leq m, 1 \leq j \leq n, 1 \leq l \leq k \leq m$ or n,
$A_{i_k} \in \mathcal{U}_i, B_{i_k} \in \mathcal{V}_j, \mathcal{U}_i = \mathcal{V}_j, u_i \in \mathcal{U}_i\}$.

3 Overview of URDS

There are two most popular modes to originate front-end of classical database systems. One is using an 'add-on' to traditional data, and it is exemplified by [13, 14, 3, 4]. The other is that fuzzy data are saved as string characters, one more step than the former to directly store the uncertain information, then weight calculation for uncertain information is considered after normal database retrieval. The latter is illustrated by [29, 18, 17, 19].

Our work belongs to the second class which is more effective though more complicated in uncertainty management. To achieve uncertainty handling, a pre-processing and post-processing system diagram was used in which pre-processing translates queries with uncertain facts into classical counterparts, and post-processing in which maps acceptable crisp queries resulting from database engine into user friendly queries by reintroducing uncertainty from libraries.

The basic design of the URDS system can be seen in the figure 1.

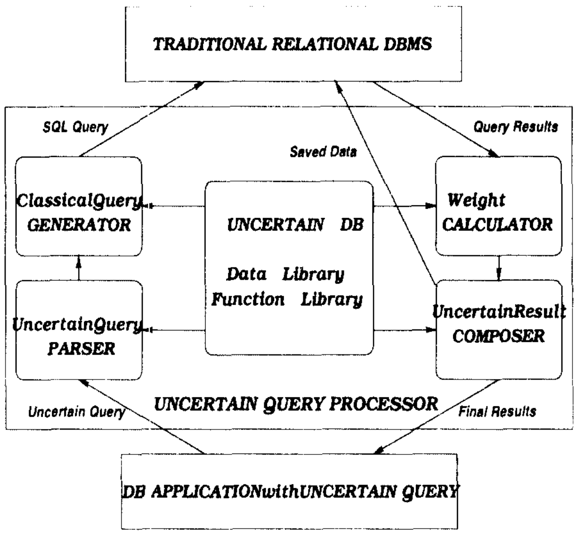

Figure 1: Architecture of URDS.

In the implementation, we pursue that the most important aspect is the simplicity, where the simplicity without losing capacity of representation and management. The URDS essentially consists of three parts: database application programs, an uncertain query processor and an ordinary relational database management system.

The database application programs are written in a host language, where we pick C language. Traditional relational DBMS as back-end processor is Oracle V7.3.2. For the middle part: uncertain query processor, we choose same host language C and embedded SQL other than OCI (Oracle Call Interface) for easy understanding the meaning of each embedded statement. Besides, they are standard and broadly used. Even there are convenient existing API calls to the OCI library. This part allows all operations with ordinary databases and is formed of following modules.

Uncertain Query Parser
compiles uncertain queries from the application programs and generates modified intermediate codes that could be linked and executed with libraries.

Classical Query Generator
produces optimum ordinary SQL codes by resolving calls from libraries to access the crisp database system and sends to DBMS engine Oracle V7.

Uncertain DB Library
primarily contains two parts: **Data library** facilitates the ad hoc definition of fuzzy membership functions, possibilistic distribution, some special values and tuple grad values between query conditions. **Function library** stores all the extensive information about relational scheme, data dictionary, relational operations and predicate evaluation.

Weight Calculator
computes all possibility distributions, probabilities, condition values given by libraries on conventional SQL values to prepare uncertain query results.

Uncertain Result Composer
formulates the results of uncertain SQL statements for the application programs from the conventional SQL ones through weight calculator. Uncertain SQL derived tables are composed from the ordinary tables plus uncertain information from libraries.

Our URD is an *intelligent database* which is clarified as a traditional relational database with additional functionalities to incorporate natural languages, some kinds of incomplete information [12, 8]. For system that will require a natural language interface to databases, dynamic adjustment of the search criteria to the underlying database will be critical.

Consequently we support that URDS has open-ended flexibility by host programs accepting and processing dynamically defined SQL statements. It is

Because of the possible complexities in programming dynamic SQL, it has been known to scare away the best of developers. But if planned and done in phases and modules, even the most complex methods can be fairly straightforward.

5 Conclusion

Fundamentally, all implementation for uncertainty management in databases can be divided into two groups: uncertainty querying in conventional databases [18, 4, 13], which is simple, and uncertainty querying in uncertain databases, which is believed in managing uncertain data and queries more competently.

We compare URDS to the systems in [18, 16] then expect that URDS has a new more user-friendly, easy-to-use and generalised querying system. It is possible to formulate queries concerning highly aggregate and vague-defined concepts which are usually of interest to human decision makers who cannot handle through conventional database systems.

Intuitively, completeness in a query language is the capability to capture all relationships. For a traditional relational database this entails the ability to form any relation by combining operations. For uncertain relational database, completeness also entails a capability to specify any partition of a domain set of formulas that is allowed by possibility and probability. Additionally, it is able to define the uncertainty level in any intermediate and resulting uncertain relations.

The implementation of URDS prototype is carried out on three steps. 1) Uncertain database data and function library are established initially without depending on other resources. 2) Weight (likelihood) calculator then classical query generator could be built on libraries. 3) Realization of parser and composer. We are now involving first and second steps partially, while the rest of them will be dealt with in forthcoming work. For that reason, performance results are not available.

References

[1] Baldwin, J. FRIL - a fuzzy relational inference language. *Fuzzy sets and systems 14* (1984), 155–174.

[2] Barbará, D., Garcia-Molina, H., and Porter, D. The management of probabilistic data. *IEEE Tran. Knowledge and Data Engineering 4*, 5 (1992), 487–502.

[3] Bosc, P., and Pivert, O. Fuzzy querying in conventional databases. In *Fuzzy Logic for the Management of Uncertainty*, L. A. Zadeh and J. Kacprzyk, Eds. John Wiley and Sons, New York, 1992, ch. 32, pp. 645–671.

[4] Bosc, P., and Pivert, O. SQLf: A relational database language for fuzzy querying. *IEEE Trans. Fuzzy Systems 3*, 1 (1995), 1–17.

[5] Buckles, B., and Petry, F. Generalized database and information systems. In *Artificial Intelligence and Decision Systems: Analysis of Fuzzy Information*, J. Bezdek, Ed., vol. III. CRC Press, Boca Raton, FL, 1986, pp. 177–201.

[6] Codd, E. Relational completeness of data base sublanguages. In *Data base system*, R. Rustin, Ed. Prentice-Hall, Englewood Cliffs, NJ, 1972, pp. 65–98. Courant computer science symposium 6, May 24-25, 1971.

[7] Codd, E. Extending the database relatonal model to capture more meaning. *ACM Trans. Database System 4*, 4 (1979), 397–434.

[8] Demolombe, R. Uncertainty in intelligent databases. In *Uncertainty Management in Information Systems*, A. Motro and P. Smets, Eds. Kluwer Academic, Boston, 1997, ch. 4, pp. 89–126.

[9] Dey, D., and Sarkar, S. A probabilistic relational model and algebra. *ACM Trans. on Database Systems 21*, 3 (sep. 1996), 339–369.

[10] Elmasri, R., and Navathe, S. *Fundamentals of database system*, 2nd ed. Benjamin/Cummings, Redwood City, CA, 1994.

[11] Fuhr, N., and Rölleke, T. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. on Information Systems 15*, 1 (Jan. 1997), 32–66.

[12] Grant, J. *Logical introduction to databases.* Harcourt Brace Jovanovich, 1987.

[13] Kacprzyk, J., and Zadrozny, S. Fquery for access: Fuzzy querying for a windows-based DBMS. In *Fuzziness in Database Management Systems*, P. Bosc and J. Kacprzyk, Eds. Physica-Verlag, Heidelberg, 1995, pp. 415–433.

[14] Kacprzyk, J., and Zadrozny, S. Fuzzy queries in microsoft access v.2. In *Proceedings of VI IFSA World Congress* (Sao Paulo, 1995), vol. 2, pp. 341–344.

[15] Lee, S. Imprecise and uncertain information in databases: An evidential approach. In *Proc. IEEE Int. Conf. Data Engineering* (1992), pp. 614–621.

[16] Medina, J., Pons, O., and Vila, M. Gefred: A generalized model of fuzzy relational databases. *Information Sciences 76* (1994), 87–109.

[17] Medina, J., Vila, M., Cubero, J., and Pons, O. Towards the implementation of a generalized fuzzy relational database model. *Fuzzy Sets and Syst. 75* (1995), 273–289.

[18] Nakajima, H., Sogoh, T., and Arao, M. Fuzzy database language and library – fuzzy extension to sql. In *Procs. of second IEEE int. conf. fuzzy systems* (Piscataway, NJ, 1993), vol. I, IEEE service center, pp. 477–482.

[19] Petry, F. E. *Fuzzy databases principles and applications.* Kluwer Academic, Norwell, Massachusetts, 1996.

[20] Pittarelli, M. Probabilistic databases for decision analysis. *Int. J. Intelligent Systems 5* (1990), 209–236.

[21] Raju, K., and Majumdar, A. K. Fuzzy functional dependencies and lossless join decomposition of fuzzy relational database systems. *ACM Trans. Database Systems 13*, 2 (June 1988), 129–166.

[22] Ramer, A., and Yu, H. Similarity, probability and database organisation. In *Proc. 1996 Asian Fuzzy System Symposium, Kenting, Taiwan* (1996), pp. 272–277.

[23] Ramer, A., and Yu, H. Semantics of dichotomous probabilistic databases. In *Proc. of International Conference on Neural Information Processing (ICONIP'97)* (Dunedin, New Zealand, 1997), IEEE NNC and INNC, pp. 564–567.

[24] Ullman, J. *Principles of Database Systems.* Computer Science Press, Rockville, MD, 1983.

[25] Umano, M. Freedom-0: A fuzzy database system. In *Fuzzy Information and Decision Processes*, M. Gupta and E. Sanchez, Eds. North-Holland, New York, 1982, pp. 339–347.

[26] Yu, H., and Ramer, A. A combined approach to uncertain data analysis. In *Advances in Intelligent Data Analysis: Reasoning about Data, Proc. of IDA97, LNCS 1280* (London, 1997), AAAI, Springer-Verlag, pp. 123–134.

[27] Yu, H., and Ramer, A. Uncertainty management in generalised relational databases. In *Procs. of European Congress on Intelligent Techniques and Soft Computing* (Aachen, Germany, 1997), vol. 2, ELITE, Verlag Mainz, pp. 1137–1141.

[28] Zadeh, L. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems 1*, 1 (1978), 3–28.

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

[30] Zimányi, E. Query evaluation in probabilistic relational databases. *Theoretical Computer Science 171* (1997), 179–219.