Efficient Updates of Uncertain Databases
Extended Abstract

Andreas Hubmer, Reinhard Pichler, Vadim Savenkov, and Sebastian Skritek

Vienna University of Technology
andreas.hubmer@alumni.tuwien.ac.at,
{pichler|savenkov|skritek}@dbai.tuwien.ac.at

1 Introduction

Uncertain databases have evolved as an active area of database research, surveyed, for example, in [10].The possible worlds semantics [1] is commonly used to deal with uncertain data. Several representation systems for uncertain databases have been proposed to provide efficient storage and query facilities for a potentially big number of possible worlds, see, e.g., [14,4,8]. Antova et al. introduced *U-relations* [2], which guarantee polynomial data complexity for queries of positive relational algebra (RA, for short). U-relations have been implemented and are available in the MayBMS system [6].

As in the case of certain databases, we clearly also want to *update* uncertain databases and pose queries of *unrestricted* RA. In [3], an API for uncertain databases, which also covers updates, is presented. The paper mentions that, for updates, decompression of the succinct representation of the possible worlds may be necessary. However, it leaves open the details and also the question if decompression could at least partly be avoided by some extension of the representation system. While the evaluation of positive RA queries on uncertain databases has been studied extensively, there has been little work beyond positive RA, like queries with *having* clauses [7] and queries with one level of *anti-join* (*not-exists* [13]). Fink et al. [5] describe an approach for unrestricted RA queries, but in a formalism that does not exhibit polynomial time data complexity for computing possible answers.

The goal of this work is to start the investigation of the update problem of U-relations and to tackle the problem of evaluating queries with anti-joins over this formalism. To this end, we will first define the anti-join operator for U-relations and show how to use it to model updates. It will turn out that the decompression of the succinct representation of possible worlds may indeed be a performance problem. We will therefore introduce an extension of the U-relation representation system and show that our new formalism may lead to an exponential decrease in the representation of an updated uncertain database. Our main contributions will be as follows.

Formal definition. We first present a formal definition of the anti-join operator on U-relations such that the result is again a U-relation. Using the anti-join, we then define the semantics of the *update operator* on U-relations.

New representations. Non-monotonous operators and updates can seriously compromise the space efficiency of U-relations. To address this problem, we introduce two generalizations of U-relations: (i) U^i-relations, which additionally allow inequalities in the descriptors of possible worlds; and (ii) U^{int}-relations, which allow intervals to specify ranges of the variables in descriptors of possible worlds. We prove that by using U^i-relations or U^{int}-relations, the required storage and the complexity of query answering can drop exponentially.

Complexity of optimization of world sets. We point out the intractability of a basic optimization problem for U-relations, U^i-relations, and U^{int}-relations.

2 Preliminaries

Propositional Logic of Finite Domains. The formalism of U-relations uses the logic known in the literature as *propositional logic of finite domains*, PLFD [12]. It is an extension of propositional logic, in which each variable x is associated with a fixed set of constants called the *domain of* x, denoted $dom(x)$. The atomic formulas in PLFD have the form $x = v$, where x is a variable and $v \in dom(x)$ is a value from the domain of x. A PLFD interpretation $(\cdot)^i$ assigns to each variable a value from its domain: $(x = v)^i = \top$ iff $x^i = v$. The interpretations of complex formulas using the usual Boolean connectives $\wedge, \vee, \rightarrow, \neg$ are defined as usual.

Possible worlds and U-relations. In the *possible worlds semantics*, an uncertain database can have multiple possible states called *possible worlds*. In the sequel, we will write P^w to denote the state of the relation P in the possible world w. Two ideas are crucial for U-relations: (i) possible worlds are described by conjunctions of PLFD atoms, and (ii) vertical decomposition of relations may allow one to exponentially reduce the size of the representation of the possible worlds.

Given an uncertain database $\mathcal{B}$ with a relational schema $\mathbf{S}$, one can find a set of PLFD variables W with accordingly chosen domains, so that each total valuation $(\cdot)^i$ assigning a value to every variable in W corresponds to exactly one possible world in $\mathcal{B}$. U-relations split each relation $R = (\overline{T}, \overline{A}) \in \mathbf{S}$ with key $\overline{T}$ in one or more *vertical partitions* $U_1, \ldots, U_m$ with the schema $(D, \overline{T}, \overline{A}_i)$, $i = 1 \ldots m$ where $\overline{A}_i$ is a subset of the attributes in $\overline{A}$, and D stores the *world-set descriptors*, or *ws-descriptors*, for short. In the original definition of the formalism of U-relations [2], the world-set descriptors in D are simply conjunctions of PLFD atoms. Since possible worlds of $\mathcal{B}$ are associated with total valuations for variables in W, each world-set descriptor ϕ defines a set of possible worlds: namely, the worlds associated with the valuations that turn ϕ to $\top$. The relation R can be recombined from the partitions $U_1, \ldots, U_m$ by joining the tuples with consistent ws-descriptors over $\overline{T}$. The semantics of the usual positive relational operators is a straightforward extension of the semantics of queries over a vertically partitioned database schema, projecting ws-descriptors along with the primary key columns: see [2] for a complete specification. The main difference is in the binary operators like cartesian product and join: they only allow combining tuples whose ws-descriptors are not contradictory. Importantly, unary

$$\text{Let } U_i := [\![Q_i]\!] \text{ with schema } [D_i, \overline{T_i}, \overline{A_i}], \text{ for } i \in \{1,2\}$$
$$\mathit{neg\_dnf\_clause}(U) := \{(d', \overline{a}) \mid (d, \overline{a}) \in U,\ d' \text{ is a clause in the DNF of } \neg d\}$$
$$\mathit{negate}(U, \overline{A}) := \mathit{neg\_dnf\_clause}\left({}_{\overline{A}}G_{\bigvee D}\left(\pi_{D,\overline{A}}(U)\right)\right).$$

$$[\![Q_1 \triangleright Q_2]\!] := \sigma_{D \not\models \bot}\left(\pi_{D_1 \wedge \overline{D}_2 \textbf{ as } D,T_1,\overline{A}_1}\left(U_1 \bowtie \mathit{negate}(U_2, \overline{A}_2)\right)\right) \cup (U_1 \triangleright U_2),$$
where the condition of anti-join $\triangleright$ and join $\bowtie$ operators is $\overline{A_1} = \overline{A_2}$.

Fig. 1: Translation of anti-join for U-relations.

operators do not alter ws-descriptors, whereas positive binary operators take a conjunction of the ws-descriptors of the relations passed as operands. Thus, positive relational operators on U-relations can be efficiently implemented.

Notation of Relational Operators. We use the extended form of the projection operator π which supports computed attributes: e.g., $\pi_{A_1+A_2 \textbf{ as } A'}(U)$ is a unary relation whose attribute A' is the sum of the attributes A_1 and A_2 in U. The *anti-join* operator ("where not exists") is denoted by the symbol $\triangleright$. The operator ${}_{\overline{A}_1}G_{f(\overline{A}_2)}(U)$ applies the aggregate function f to the set of attributes $\overline{A}_2$ of U, whereby the tuples of U are grouped by the values of attributes $\overline{A}_1$.

3 Anti-joins and updates on U-Relations

As pointed out in the previous section, U-relations allow for a natural implementation of positive relational operators. However, the non-monotonic operators over U-relations have not yet been considered. One such operator especially well-suited for U-relations, which must contain tuple ids, is the anti-join.

Definition 1. *The* anti-join $R \triangleright P$ *is the uncertain relation defined, for each possible world* w*, by the expression* $R^w \setminus (R^w \ltimes P^w)$.

Theorem 1. *The translation in Fig. 1 satisfies Definition 1.*

Example 1. Consider the U-relation $U = (D, T, A)$ consisting of the following tuples: $\langle x = 1, t_1, a\rangle, \langle x = 2, t_1, b\rangle, \langle y = 1, t_2, a\rangle, \langle y = 2, t_2, c\rangle, \langle y = 3, t_3, a\rangle$. The result $U_\triangleright$ of the relational expression $\sigma_{T=t_1}(U) \triangleright_A \sigma_{T\neq t_1}(U)$ is found as follows. $U_\triangleright$ contains all possible t_1-tuples of U that do not match any possible $t_{2,3}$-tuples. This corresponds to the $(U_1 \triangleright U_2)$ branch in Fig. 1. Such a tuple is $\langle x = 2, t_1, b\rangle$. Additionally, for t_1-tuples that have matches in t_2 or t_3, the transformation of ws-descriptors using *negate* is needed. Such tuple is $\langle x = 1, t_1, a\rangle$. The call $\mathit{negate}(\sigma_{T\neq t_1}(U), A)$ first aggregates ws-descriptors of tuples that satisfy the condition $A = a$ using $\vee$ and then passes the relation $\langle y = 1 \vee y = 3, a\rangle$ to *neg_dnf_clause*. The latter function negates the aggregate ws-descriptor and transforms it to the DNF: provided that $dom(y) = \{1, 2, 3\}$, the resulting DNF has a single clause $y = 2$. Thus, the result of $\mathit{negate}(\sigma_{T\neq t_1}(U), A)$ is $\{\langle y = 2, a\rangle\}$.

The join $(\sigma_{T=t_1}(U) \bowtie_A negate(\sigma_{T\neq t_1}(U), A))$ followed by merging ws-descriptors with $\wedge$ thus gives $\langle x = 1 \wedge y = 2, t_1, a\rangle$. Since $(x = 1 \wedge y = 2) \not\models \bot$ holds, the result of the anti-join query $U_\triangleright$ is $\{\langle x = 1 \wedge y = 2, t_1, a\rangle, \langle x = 1, t_1, a\rangle\}$. □

Important on its own right, the anti-join operator allows implementing updates of U-relations. Consider a general form GU of the update operator:

$$GU\colon\ \textbf{update}\ R\ \textbf{set}\ A_k = Q.Expr\ \textbf{from}\ Q \bowtie R$$

Such update queries are supported by all major RDBMS's. GU updates an attribute A_k of a relation $R = (A_1, \ldots, A_k, \ldots, A_n)$ and allows to join R with some query Q serving as a source for the new value $Expr$ of A_k. It is essential that the update be unambiguous, i.e. if a tuple in R has more than one join partner in Q then *Expr* must return the same value for all join partners. The update GU can be expressed as a query Q_{GU}, which is a union of two queries, in which the former selects the updated tuples and the latter selects the unchanged ones:

$$Q_{GU}\colon\ \pi_{A_1,\ldots A_{k-1},Expr,A_{k+1},\ldots,A_n}(R \bowtie Q) \cup (R \triangleright Q)$$

It is not difficult to show that no positive relational expression can express Q_{GU}.

4 Extended U-relations

4.1 Ws-descriptors with inequalities and intervals

As follows from the definition in Fig. 1 and from Example 1, the performance of anti-join on U-relations depends crucially on the function *negate*. In compliance with the definition of U-relations [2], this function produces ws-descriptors being conjunctions of PLFD atoms. In the presence of non-monotonic queries or updates such restricted form of ws-descriptors may prevent U-relations from delivering on the promise of representing possible worlds succinctly.

Example 2. Consider a schema $R = (T, A_1, \ldots, A_n)$, where T is the key attribute, split into n U-relations $U_i = (D, T, A_i)$. Let $\mathcal{B}$ be an uncertain instance of relation R, and let the tuple t in R have possible values constructed by picking an arbitrary value from some fixed set of constants V for each attribute $t.A_i$. Using the vertical partitioning in U-relations, the $|V|^n$ possible values of t can be represented by total $n \cdot |V|$ tuples in the U-relations $U_i, \ldots, U_n$. E.g., for $n = 3$ and $V = \{a, b, c\}$, the following tuples in the U-relations are needed. U_1 : $\langle x = 1, t, a\rangle$, $\langle x = 2, t, b\rangle, \langle x = 3, t, c\rangle$, U_2 : $\langle y = 1, t, a\rangle, \langle y = 2, t, b\rangle, \langle y = 3, t, c\rangle$, $U_3 : \langle z = 1, t, a\rangle, \langle z = 2, t, b\rangle, \langle z = 3, t, c\rangle$.

Now, consider the result of the relational expression $\mathcal{B} \triangleright \{R(t, a, a, a)\}$. The entries with $T = t$ in the U-relations should represent the remaining 26 possible values of t: $V \times V \times V \setminus \{a, a, a\}$. After recombining R from $U_{1,2,3}$ and subtracting $\{\langle x = 1 \wedge y = 1 \wedge z = 1, t, a, a, a\rangle\}$ according to the definition of $\triangleright$, U_1 contains tuples $\langle x = 1 \wedge y = 2 \wedge z = 1, t, a\rangle$, $\langle x = 1 \wedge y = 3 \wedge z = 1, t, a\rangle$, $\langle x = 1 \wedge y = 1 \wedge z = 2,\ t, a\rangle$, $\langle x = 1 \wedge y = 1 \wedge z = 3, t,\ a\rangle$, $\langle x = 2, t, b\rangle, \langle x = 3, t, c\rangle$. U_2 and

U_3 need to be updated accordingly. That is, 6 tuples with $T = t$ per U-relation are now needed to represent the result of anti-join operator, instead of the 3 tuples per U-relation in $\mathcal{B}$. One can show that for R with n attributes split into n U-relations, additional $m \cdot (|V|^n - |V|)$ entries in each U-relation are needed to represent the deletion of one possible world with m tuples from $\mathcal{B}$. □

We address this issue by allowing more expressive ws-descriptors.

• *Iws-descriptors* allow for inequalities along with equalities to be present in the conjunctive formulas. Thereby the PLFD formula $x \neq v_i$ is a shorthand for $(x = v_1) \vee \ldots \vee (x = v_{i-1}) \vee (x = v_{i+1}) \vee \ldots \vee (x = v_m)$, where $dom(x) = \{v_1, \ldots, v_m\}$, which explains the source of increased space-efficiency of the new representation. The resulting formalism is called *U^i-relations* .

• *Intws-descriptors* replace equalities and inequalities with intervals of values which the given variable can take. An intws-descriptor is a set of triples (v, lo, hi) that consist of a variable v with a lower bound lo and an upper bound hi, such that $lo \leq hi$, $lo, hi \in dom(v)$ and such that each v occurs at most once in the set. It is assumed that the domain of v is $0 \ldots n$, for some integer n. The respective modification of U-relations is called *U^{int}-relations.* For instance, for $v_i \notin \{0, n\}$, we can represent the formula $x \neq v_i$ by two triples $(x, 0, v_i - 1)$ and $(x, v_i + 1, n)$.

Theorem 2. *There exist ws-descriptors whose negation represented as sets of iws- resp. intws-descriptors is exponentially more succinct than if represented as sets of ws-descriptors.*

For instance, using U^i-relations the result of the anti-join $\mathcal{B} \triangleright \{R(t, a, a, a)\}$ from Example 2 can be expressed by exactly the same number of tuples as in $\mathcal{B}$:

$U_1 : \langle x = 1 \wedge y \neq 1, t, a\rangle, \langle x = 2, t, b\rangle, \langle x = 3, t, c\rangle,$
$U_2 : \langle y = 1 \wedge x \neq 1 \wedge z \neq 1, t, a\rangle, \langle y = 2, t, b\rangle, \langle y = 3, t, c\rangle,$
$U_3 : \langle z = 1 \wedge x \neq 1 \wedge y \neq 1, t, a\rangle, \langle z = 2, t, b\rangle, \langle z = 3, t, c\rangle.$

Extending ws-descriptors preserves the favorable properties of U-relations:

Theorem 3. *Positive relational algebra queries can be evaluated on U^i- resp. U^{int}-relational databases in polynomial time data complexity.*

Proof (Sketch). Evaluating positive RA operators on extended U-relations only differs from evaluating these operators on U-relations of [2] in the consistency test of the binary operators: the tuples in two relations are only combined if the respective ws-descriptors are mutually consistent. That is, if the conjunction of two ws-descriptors is satisfiable. The satisfiability test is feasible in polynomial time in all three cases of U-relations, U^i-relations and U^{int}-relations. Therefore, the claim of the theorem follows from the result in [2] claiming that the problem of evaluating positive RA operators on U-relations has polynomial data complexity. □

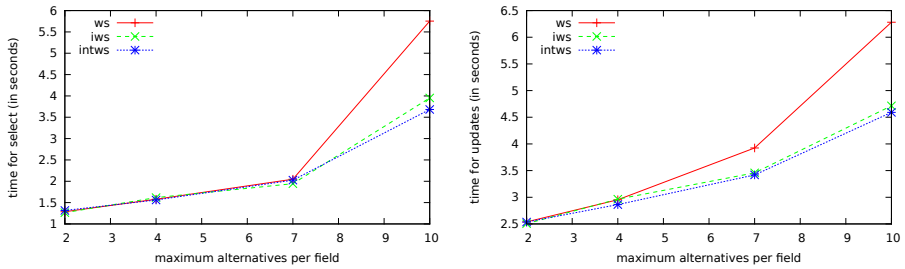

Fig. 2: Performance impact of the extensions of U-relations.

4.2 Optimality of ws-descriptors

The performance of queries and updates depends on the choice of the concrete expression representing the set of possible worlds. Below, we show that optimizing sets of ws-descriptors is intractable even without the extensions proposed above. Obviously, this intractability result also holds for U^i-relations and U^{int}-relations.

Proposition 1. *Let S be a set of ws-descriptors occurring in a given U-relation U. Let $\omega(S)$ denote a set of all valuations which turn at least one ws-descriptor in S to $\top$. It is Σ_2^P-complete to decide whether a ws-descriptor S' exists, such that $|S'| < |S|$ and $\omega(S) = \omega(S')$.*

Proof (Sketch). We show hardness by a reduction from the DNF minimization problem Min-DNF(ϕ, k) [9] defined as follows:

Given a propositional formula ϕ in DNF and an integer k, is there a DNF formula equivalent to ϕ that contains k or fewer occurrences of literals?

This problem has been shown to be Σ_2^P-complete by Umans [11]. We encode an instance of Min-DNF(ϕ, k) as a problem in the claim of the theorem. Thereby each propositional variable x in ϕ gives rise to a fresh PLFD variable $\dot{x}$ with domain $\{1, 2\}$, and each conjunctive clause in ϕ is encoded into a ws-descriptor by replacing the positive propositional literal x with the PLFD atom $\dot{x} = 1$ and each negative literal $\overline{x}$ with the PLFD atom $\dot{x} = 2$.

For membership, consider the following algorithm: guess a set of ws-descriptors S' and check in polynomial time whether it contains at most k equalities (PLFD atoms). With a coNP oracle we check that there does not exist a valuation w such that $w \in \omega(S)$ and $w \notin \omega(S')$ or vice versa. The check itself is feasible in polynomial time. □

5 Conclusion

In this paper we have introduced two features of U-relations missing so far: the non-monotonous operator of anti-join and updates. Moreover, we have shown

how the formalism of U-relations can be extended in order to improve the performance of these operations. We have also implemented our extensions on top of the open-source MayBMS system [6] and experimentally verified their efficiency. For example, Fig. 2 shows the impact of extended U-relations from Section 4 on the performance of both select and update queries, for various limits of possible values which uncertain tuple attributes can take.

To deal with the high worst-case complexity of optimizing U-relations, we have incorporated some easy heuristics that often lead to better representations (which are not guaranteed to be optimal, though). Our experiments show the effectiveness of these heuristics. A systematic investigation of such methods and further experiments with extending U-relations, enabling efficient support of updates and non-monotonic operators, remains a subject of future research.

Acknowledgements. This work has been supported by the Austrian Science Fund (FWF): P25207-N23.

References

1. Serge Abiteboul, Paris Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.*, 78:159–187, 1991.
2. Lyublena Antova, Thomas Jansen, Christoph Koch, and Dan Olteanu. Fast and simple relational processing of uncertain data. In *Proc. of ICDE '08*, pages 983–992. IEEE, 2008.
3. Lyublena Antova and Christoph Koch. On APIs for probabilistic databases. In *Proc. of QDB/MUD '08*, pages 41–56, 2008.
4. Jihad Boulos, Nilesh N. Dalvi, Bhushan Mandhani, Shobhit Mathur, Christopher Ré, and Dan Suciu. MYSTIQ: A system for finding more answers by using probabilities. In *Proc. of SIGMOD '05*, pages 891–893. ACM, 2005.
5. Robert Fink, Dan Olteanu, and Swaroop Rath. Providing support for full relational algebra in probabilistic databases. In *Proc. of ICDE '11*, pages 315–326. IEEE.
6. Christoph Koch. MayBMS: A system for managing large uncertain and probabilistic databases. In *Managing and Mining Uncertain Data*. Springer, 2008.
7. Christopher Ré and Dan Suciu. The trichotomy of having queries on a probabilistic database. *The VLDB Journal*, 18:1091–1116, 2009.
8. Sarvjeet Singh, Chris Mayfield, Sagar Mittal, Sunil Prabhakar, Susanne Hambrusch, and Rahul Shah. Orion 2.0: native support for uncertain data. In *Proc. of SIGMOD '08*, pages 1239–1242. ACM, 2008.
9. Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.
10. Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Morgan & Claypool, 2011.
11. Christopher Umans. The minimum equivalent DNF problem and shortest implicants. In *Proc. of FOCS '98*, pages 556–563. IEEE, 1998.
12. Andrey Voronkov. Propositional logic of finite domains. Chapter 12 of the "Logic and Modelling" course script, University of Manchester. `http://www.voronkov.com/lm_doc.cgi?what=chapter&n=12`, Accessed: March 22, 2013.
13. Ting-You Wang, Christopher Ré, and Dan Suciu. Implementing NOT EXISTS predicates over a probabilistic database. In *Proc. of QDB/MUD '08*, pages 73–86.
14. Jennifer Widom. Trio: A system for data, uncertainty, and lineage. In *Managing and Mining Uncertain Data*. Springer, 2008.