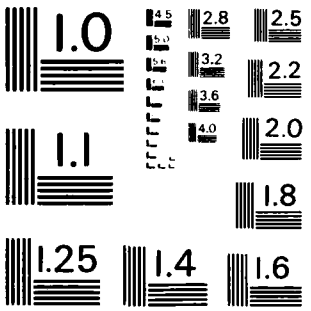
AD-A135 707 UNIVERSAL RELATION DATABASE SYSTEMS(U) STANFORD UNIV CA 1/1
DEPT OF COMPUTER SCIENCE J D ULLMAN AUG 83
AFOSR-TR-83-0962 AFOSR-80-0212

UNCLASSIFIED F/G 5/2 NL



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(2) (1)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0962	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) UNIVERSAL RELATION DATABASE SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED Annual, 1 SEP 82-31 AUG 83
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jeffrey D. Ullman		8. CONTRACT OR GRANT NUMBER(s) AFOSR-80-0212
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science Stanford University Stanford CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research /NM Bolling AFB DC 20332		12. REPORT DATE Aug 83 13. NUMBER OF PAGES 5
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The query facility for their universal relation database system is now working. The fundamental paper unifying ideas on what a UR system can and should be has been published. A paper surveying developments in the field of universal relation systems was invited for the triennial IFIP Congress and was delivered in September. Some initial results on logical theories applied to the problem of updating views have been obtained. There have been a number of developments concerning inference of inclusion dependencies and on the complexity of deciding certain properties of database schemes. Some interesting (CONTINUED)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ITEM #20, CONTINUED: results on the difficulty of obtaining hash functions that work well for particular sets of data have been obtained and won an award.

ANNUAL REPORT FOR AFOSR [redacted] 80-0212
UNIVERSAL RELATION DATABASE SYSTEMS
Sept. 1, 1982 Aug. 31, 1983

SUMMARY

We now have the query facility for our universal relation database system working. We have published what we believe to be the fundamental paper unifying ideas on what a UR system can and should be. A paper surveying developments in the field of universal relation systems was invited for the triennial IFIP Congress and was delivered in September. Some initial results on logical theories applied to the problem of updating views have been obtained. There have been a number of developments concerning inference of inclusion dependencies and on the complexity of deciding certain properties of database schemes. Some interesting results on the difficulty of obtaining hash functions that work well for particular sets of data have been obtained and won an award.

I. System/U

We now have a working version of "System/U," our experimental universal-relation query answering system. A translator of queries to parse trees, based on our view of universal relation semantics, has been working since last summer, and during the past year we completed the optimization phase that translates these trees into an ordered set of steps that implements the query efficiently. The final stage, where the optimized sequence of steps is executed on files that store the actual database relations, was implemented on top of ERIS, which is Steve Reiss's (Brown Univ.) relational database facility.

A description of the system, its data definition language, its query language, and the important algorithmic ideas used to implement them so far has been compiled [KKU] and submitted for publication.

II. Universal Relation Semantics

The paper [MUV1] was published in a recent conference proceedings, and an expanded version [MUV2] has been accepted for *TODS*. These works unify the various assumptions that people have suggested were necessary to make universal relation systems work. We identified a basic assumption, called the *one-flavor assumption*, that we believe is essential for a database scheme to allow meaningful UR queries, and we believe that this condition is also sufficient. Briefly, we require that two tuples in the relation produced by a UR system be interchangeable, in the sense that if there are several paths that might give rise to tuples, the user doesn't care which path was actually taken to produce a tuple. Armed with this viewpoint, the database designer can decide when attributes must be split, and when cycles can be permitted in the scheme. Of course, the designer's judgement is needed to decide when the one-flavor assumption is violated, but design judgement is always needed, whatever the framework in which a database is designed.

Beyond the fundamental assumptions like the one-flavor assumption, there are two viewpoints people have taken to define the "correct" response of a UR system.

1. Give an algorithm for interpreting queries. System/U is an example of this approach.
2. Define an abstract universal relation, and require that the response by the system be as if the query were applied to this one relation. The work of Sagiv† is an example of this approach.

We showed in [MUV1, MUV2] the following equivalence between these two viewpoints. If there is any first order way (i.e., a formula in relational algebra) to produce the result of applying the query to the

† "Can we use the universal instance without using nulls," *ACM SIGMOD International Symposium on Management of Data*, pp. 108–120, 1981.

representative instance (Sagiv's [S] definition of the abstract UR), then we can do so by a finite union of lossless "tableau mappings." The latter are essentially expressions using lossless joins and projections. Interestingly, the System/U approach, which produces unions of lossless joins, is not so far from the most general possible approach, although we now see that there are situations where we shall miss generating certain tuples that might logically deserve being generated.

We are beginning to see the first papers on UR semantics that were written with support of the grant appearing in journals. Recently, the papers [FMU] and [MU] were published. The ideas behind universal relation systems, after having endured many years of often outright hostility, are finally being recognized as significant. The paper [U] was invited to the 1983 IFIP Congress, and a rebuttal written by Ullman to a logically invalid attack on the UR concept that appeared in *TODS* in 1981 is finally to appear in the next issue of that journal.

III. Logical Databases and Updates to Views

The problem of implementing updates to views, of which the universal relation is a special case, has received considerable attention recently. We believe that general schemes for accepting update requests about fictional relations and translating them in an understandable and justifiable way to updates on the actual relations can only be developed after one has an understanding of what the "meaning" of the update is. We have therefore begun consideration of logical theories as sets of facts that are (explicitly or implicitly) found in the database.

In [FUV] we set down a viewpoint in which databases are sets of facts, presumably including the facts stored in the database, and possibly including some facts constructible from those facts and present in one or more views. We also proposed a particular viewpoint regarding how an insertion or deletion affects the set of facts in the database. First, when deleting a fact, the fact should no longer be implied by the facts in the database, a point that seems incontestable. Next, when inserting a fact, the fact should then be in (not just implied by) the database, and the database should not imply the negation of the fact, again a very reasonable point to take. Third, we wish to assume that the database change is minimal, in the sense that we do not delete a fact that could just as well be left in without contradicting anything, and we do not insert spurious facts.

The major debatable assumption we make is that we do not delete any facts unless absolutely forced to, and only as a second priority do we minimize the number of extra facts inserted. We are not wedded to this point of view, but we like it on the grounds that the database represents facts that the users believe, and we must be very careful about throwing them away. On the other hand, new facts, we show, need only be inserted in response to an insertion request by the user.

Some interesting consequences follow from our assumptions. First, we discovered that it is essential to make a distinction between the actual facts in the database, and the *closure* of the database, i.e., the facts that follow logically from those in the database. This, in turn, means that two theories, i.e., database states, can be logically equivalent, in the sense that each statement in one follows from statements in the other, and yet have different properties under insertion and deletion.

Example 1: The theories $T_1 = \{a, b\}$ and $T_2 = \{a, b, a \vee b\}$ are logically equivalent, since $a \vee b$ follows logically from the statements a and b in T_1, and all other members of either theory are present in the other. However, if we delete a and then delete b, from both theories, T_1 becomes empty, while T_2 retains $a \vee b$, so the two theories become logically inequivalent.

It may seem bizarre that $a \vee b$ can remain true after we deleted a and we deleted b. However, this example points up several facts.

1. The meaning of a deletion is only that we are no longer sure the fact is true. If we wanted to say that a is absolutely not true, we would insert $\neg a$, which is different from deleting a.
2. Certain derived facts, like $a \vee b$ apparently play the role of an integrity constraint, i.e., it is saying that at all times, either a is true or b is true. We must not put them in the database explicitly unless we mean them.
3. As a generalization of (2), the details of what statements are in the database matters, even if the statements in question are logical consequences of others in the database.

The second major point from [FUV] is that deletions, rather than insertions, are central, since is shown that insertion of statement s is equivalent to deletion of $\neg s$ followed by adjoining s to the database. In general, there will be more than one theory that can be obtained by a minimal change, since several changes may be incommensurate.

Example 2: Suppose our database includes Employee-Department facts and Department-Manager facts, and there is an Employee-Manager view formed from the latter two relations by the obvious composition. Then if Jones is in the Toy Dept., and that department is managed by Smith, there is the derived fact that Smith manages Jones, which may or may not be explicitly in the theory, depending on our specific update rules. If someone says "delete the fact that Smith manages Jones," then we are left with two different minimal changes:

1. Delete the fact that Jones is in the Toy Dept.
2. Delete the fact that Smith manages the Toy Dept.

The viewpoint taken by [FUV] is that the actual database should be adjusted to be logically equivalent to the "or" of these two possible worlds. That is, the new database consists of the single fact

"Jones is in the Toy Dept. or Smith manages the Toy Dept."

assuming there were no other facts to begin with. Notice how we hold onto the strongest fact that does not imply the deleted fact, yet can be built from facts formerly in the database.

Example 3: As an example of the [FUV] rule for combining theories, let $T_1 = \{a, b\}$ and $T_2 = \{a, c, d\}$. Then $T_1 \vee T_2 = \{a, a \vee c, a \vee d, b \vee c, b \vee d\}$. Note that statements like $a \vee b$ are kept in, even though implied by the statement a, which is really $a \vee a$, the two a's being taken from the two theories. This aspect of the definition is essential for certain results to go through, it seems.

IV. Complexity Issues and Dependency Theory

There has been significant progress in the development of algorithms for reasoning about functional dependencies and inclusion dependencies. The latter are dependencies that say an entry in one or more columns of one relation must also appear in designated columns of another (perhaps the same) relation. Typical constraints of this form are "every Manager is an Employee", or "if the department d is mentioned in the EMPS relation, then there is also a tuple for d in the DEPTS relation." Functional and inclusion dependencies are by far the most common forms of dependencies found in practice, yet while the former are well understood, the latter have been largely ignored by the theory, and their interaction has been unknown.

In [KCV], the interaction between FD's and unary inclusion dependencies (those involving the containment of a single attribute in another - by far the most common case) was uncovered, and an efficient algo-

rithm for deducing all consequences was given. However, for FD's and binary inclusion dependencies, there is no algorithm to find the consequences [CV].

Further explorations have been made into a number of other areas of dependency theory. [GMV] discusses what it means for databases (sets of relations) to satisfy dependencies. This issue is of importance for UR semantics, because we presume that any (imaginary) universal relation will satisfy given dependencies in some sense. [GV] shows how to test consistency, one of the notions from [GMV], in polynomial time, provided the dependencies are "total," i.e., they apply to the UR as a whole.

V. Properties of Acyclic Database Schemes

We have long felt that the "acyclic" database schemes played a fundamental role in design of databases. In past years we reported a large number of useful properties possessed by these schemes but not by cyclic ones. The basic query answering strategy of System/U depends on decomposing the database scheme into acyclic subschemes.

The paper [BV] shows another useful property of acyclic schemes. Specifically, for these schemes (but not in general), the natural join is the way to reconstruct the universal relation whenever a unique universal relation exists. This is further confirmation that we have adopted the correct approach to answering queries, since we are assured that our system will construct the universal relation correctly and answer the query as if it were asked about that universal relation.

VI. Efficient Retrieval

In [M], which won the Machtey Prize for the best student paper at the upcoming IEEE Symp. on Foundations of Computer Science, the issue discussed is the tradeoff between the performance of hashing functions (how many collisions they induce on particular sets of data) and the *program complexity* of the functions, i.e., how long the program computing the hash function must be. A variety of hashing-like schemes are discussed, including hashing with secondary chains and hashing into blocks.

In addition to getting precise bounds on how long the typical program must be for each of these schemes, the paper and doctoral thesis concludes that there is no advantage to a scheme in which pointers are used to form chains, when compared with schemes that calculate addresses directly. The best sort of scheme appears to be one where the hash function gives the address of a block of memory where the datum in question may be stored, and binary search within the block is used to find the datum if it exists. This observation about pointers appears to be borne out in practice, even when the data set is changing rather than fixed.

Publications Supported by Grant

[BV] C. Beeri and M. Y. Vardi, "On acyclic database decompositions," STAN CS 83 976, Stanford CSD, July, 1983.

[CV] A. K. Chandra and M. Y. Vardi, "The implication problem for functional and inclusion dependencies is undecidable," unpublished memorandum, Stanford University, Stanford CA, 1983.

[FMU] R. Fagin, A. O. Mendelzon, and J. D. Ullman, "A simplified universal relation assumption and its properties," *ACM Transactions on Database Systems* 7:3 (Nov., 1982), pp. 343 360.

[FUV] R. Fagin, J. D. Ullman, and M. Y. Vardi, "On the semantics of updates in databases," *Proc. Second ACM Symposium on Principles of Database Systems*, pp. 352 365, March, 1983.

[GMV] M. H. Graham, A. O. Mendelzon, and M. Y Vardi, "Notions of dependency satisfaction," STAN

CS 83 979, Stanford CSD, Aug., 1983.

[GV] M. H. Graham and M. Y. Vardi, "On complexity and axiomatizability of consistent database states," unpublished memorandum, Stanford Univ., 1983.

[KCV] P. C. Kanellakis, S. C. Cosmodakis, and M. Y. Vardi, "Unary inclusion dependencies have polynomial time inference problems," *Proc. Fifteenth Annual ACM Symposium on the Theory of Computing*, pp. 264 277, 1983.

[KKU] H. F. Korth, G. M. Kuper, and J. D. Ullman, "System/U: a database system based on the universal relation assumption," STAN CS 82 944, Dept. of CS, Stanford Univ., Jan., 1983.

[M] H. G. Mairson, "The program complexity of searching a table," to appear in *Proc. Twenty-fourth IEEE Symp. on Foundations of Computer Science*, Nov., 1983.

[MU] D. Maier and J. D. Ullman, "Maximal objects and the semantics of universal relation databases," *ACM Transactions on Database Systems* 8:1 (March, 1983), pp. 1 14.

[MUV1] D. Maier, J. D. Ullman, and M. Y. Vardi, "The return of the JD," *Proc. Second ACM Symposium on Principles of Database Systems*, pp. 279 287, March, 1983.

[MUV2] D. Maier, J. D. Ullman, and M. Y. Vardi, "The equivalence of universal relation definitions," STAN CS 82 940, Dept. of CS, Stanford Univ., Oct., 1982. To appear in *TODS*.

[MV] J. A. Makowsky and M. Y. Vardi, "On the expressive power of data dependencies," unpublished memorandum, Stanford Univ., 1983.

[U] J. D. Ullman, "Universal relation interfaces for database systems," *Proc. 1983 IFIP Congress*, pp. 273 282, North Holland, Amsterdam.

Ph. D. Thesis Supported by Grant

Harry G. Mairson, "The Program Complexity of Searching a Table," approx. Dec., 1983.

Personnel Supported by Grant

Faculty:		
	J. D. Ullman	25%, Oct., 1982 June, 1983
Consultant:		
	D. Maier	5 days
Postdoc:		
	M. Y. Vardi	15%, Sept., 1982 Aug., 1983
Research Assistants:		
	J. Feigenbaum	25%, Oct.-Dec., 1982 50%, Jan.-June, 1983
	G. Kuper	100%, Sept., 1982 50%, Oct., 1982 June, 1983 100%, July Aug., 1983
	H. G. Mairson	50%, Oct., 1982 June, 1983 100%, July Aug., 1983
	A. Van Gelder	100%, Sept., 1982 50%, Oct. Dec., 1982 25%, Jan. March, 1983 50%, April Aug., 1983


DA
FILM