The research carried out under this grant from 1979-1984 centered around the design of universal relation database systems. Certain theoretical aspects of dependency theory received attention, especially the theory of acyclic hypergraphs and their corresponding join dependencies. We investigated the use of first-order logic as a way to describe the effect of updates on universal-relation databases, and views in general. A number of other topics concerning database systems also received attention, including concurrency control by locking, data coding for massive write-once memories, hash table design, and logical interfaces for database systems.
AFOSR-TR- 35-0673

Final Report
Grant AFOSR 80-0212
15 June 1980 - 1 November 1984

Title
DATABASE THEORY

Principal Investigator
J.D. Ullman
Stanford University
Stanford, CA 94305

Accession For
NTIS GRAI
DTIC TAB
Unannounced
Justification

By
Distribution/
Availability Codes
Dist
Avail and/or
Special

Approved for public release; distribution unlimited.
Summary
The research carried out under this grant from 1979–1984 centered around the design of universal relation database systems. Certain theoretical aspects of dependency theory received attention, especially the theory of acyclic hypergraphs and their corresponding join dependencies. We investigated the use of first-order logic as a way to describe the effect of updates on universal-relation databases, and views in general. A number of other topics concerning database systems also received attention, including concurrency control by locking, data coding for massive write-once memories, hash table design, and logical interfaces for database systems.

System/U
The central objective of a universal-relation system is to relieve the user of the need to know system details that he is required to know in order to use conventional systems. An extreme example of what we might hope to achieve is a natural language interface, where one could say something like

"Tell me all you know about the Zanabian X-27 bomber."

But natural language systems are not feasible at the moment, and may never be precise enough for unambiguous retrieval of information.

In the "System/U" project we have tried to take a middle approach, implementing a formal query language, but a language that was better tuned to what a nonspecialist user might know about the database. Thus, we would ask in the System/U query language:

retrieve Specifications
where Country = "Zanabia" and Designation = "X-27"

In comparison, a typical relational database system, which are the easiest-to-use commercial systems, might state the same query this way.

range of A is AIRCRAFT
range of D is DATA
retrieve (D.Specifications)
where A.Country = "Zanabiae and A.Designation = "X-27"

The difference is that the user of the conventional system not only has to know about kinds of data (Specifications, Countries, and Designations), but about how that data is organized (into relations AIRCRAFT and DATA), and how the data in the two relations is logically connected (in the above example, the connection is by fields in both relations, named "Aircraft-ID").

System/U is an example of a "universal relation" system, in which the user may
imagine that the data is stored in a large table, with one column for each of the "attributes," such as Specifications, Countries, and Designations, above. The principal implementation problem is how to make inferences about what connections are implied by the user's query mentioning some particular subset of the attributes. In the above example, the connection would be easy to infer, but real systems may have hundreds of attributes, connected in roundabout ways, not all of which are intuitively meaningful. A brief summary of our approach to this problem is:

1. Data dependencies, which are simple predicate logic statements about constraints on data (e.g., no country has two planes with the same designation), are used to infer what connections among attributes are meaningful. These dependencies are declared once, when the database is designed, and must reflect the user's intuition but need not be known to the user.

2. To test the validity of an algorithm for answering queries, we must have a model of the universal relation that the user "sees." Such a relation can be defined from the dependencies and the data in the actual database relations, in one of several ways.

3. Query-answering algorithms produce answers by applying algebraic operations to the database itself. Ideally, we would like the query result to be the same as the user would have gotten if the query were applied directly to the (imaginary) universal relation.

4. Mathematical developments in universal relation theory tell us when a database structure allows us to find an algebraic way to simulate queries on the universal relation (which we cannot construct in practice because it is too large), and in many cases give us algorithms for translating queries about the universal relation to queries on the real database.

The important features of System/U include:

a) The structure is defined by a data definition language, in which the user states data dependencies and "objects," which are the indivisible groupings of attributes. In the above example, (Aircraft-ID, Country) and (Aircraft-ID, Designation) might be objects, but objects of more than two attributes are not uncommon.

b) The system deduces from this information "maximal objects," which are maximal collections of objects within which connections make sense according to the given dependencies. The database designer may alter the system's selection, but once set, all queries are interpreted using these maximal objects, and the user does not have to know about them.

c) A query about a set of attributes is answered by taking the union of the connections within all maximal objects that include all the attributes of the query. Maximal objects are always acyclic hypergraphs in a special sense that we developed for the purpose, and which is a natural generalization of a tree (although the standard definition of "acyclic hypergraph" disagrees with ours and is not suitable for our applications). The desired connection is the unique minimal subtree containing all the attributes in the query.

The primary source for the description of the system and the algorithms underlying it is Korth et al. [1984]. The "maximal object" idea is from Maier and Ullman [1983]. Beeri and Korth [1982] described some early ideas on the design of the system. The ideas found in System/U engendered a debate in the database community concerning the ability
of universal relation systems to represent real-world data correctly. Ullman [1982, 1983a] were two attempts to refute erroneous reasoning that had appeared in print, and Ullman [1983b] is an invited paper on the universal relation ideas presented to the international community.

Copies of the system have been sent to two other sites on an experimental basis. More importantly, one can see the ideas developed in the System/U project appearing in other systems, both experimental and commercial.

Acyclic Hypergraphs

One of the first things we discovered about universal relation systems is that they were much easier to implement, and the answers obtained from them made much more sense if the objects defining the database had a structure we called "acyclic." The study of hypergraphs with this property was therefore a major effort under the grant. The original ideas were from Fagin, Mendelzon, and Ullman [1982]. The properties of acyclic hypergraphs were explored in Beeri et al. [1981], Beeri and Vardi [1983], and Maier and Ullman [1984].

Adequacy of Database Structures

The property of acyclicity interacts with data dependencies, so certain cyclic object structures in a database scheme may in fact allow queries to be posed on a universal relation and still make sense. The key issue is the existence of "weak instances," which are universal relations whose projections onto the actual database relations are (not necessarily proper) supersets of what actually appears there. Maier, Ullman, and Vardi [1984] answered the question of when queries on a universal relation can be answered by algebraic operations on the database. Graham, Mendelzon, and Vardi [1983], Graham and Vardi [1984], and Vardi [1982] dealt with the question of when universal relation queries "made sense."

Dependency Theory

Questions of whether a universal relation exists for a given database structure depends not only on the structure, but on what dependencies are assumed. A "dependency" in its most general sense is any constraint on what values the relations in the database may assume. "Template dependencies," which are far more general than dependencies like "functional" and "multivalued" studied earlier, were investigated by Fagin, Maier, Ullman, and Yannakakis [1983]. Interestingly, template dependencies later turned out to have a central role in characterizing when algebraic answers to queries existed, as was discussed in Maier, Vardi, and Ullman [1984].

Another important class of dependencies, the inclusion dependencies, were studied by Kanellakis et al. [1983]. These dependencies say that a value appearing in one context must also appear in another context; e.g., a name appearing as a manager must also appear as an employee.

Write-Once Memory

Mass data storage on optical disks is an exciting future possibility, but they present a number of unusual problems, because 1's can never be overwritten by 0's. In Dolev et al. [1984]
we studied the ways in which errors, which appear on the disk because of imperfections in the medium, could be corrected.

Updates

While we believe the techniques for designing and querying universal relation databases are adequate, little progress has been made in answering the question of when an update expressed in terms of the universal relation makes sense in terms of the underlying database. In fact, researchers concerned with conventional database systems have a similar problem, that of updating through views. In Fagin, Kuper, Ullman, and Vardi [1984] and in Kuper, Ullman, and Vardi [1984] we considered the contents of the database to be a “flock” of logical theories, i.e., a set of sets of first-order statements. We then investigated how insertions and deletions of facts affect these theories and cause them to split into two or more theories.

More conventional problems of updates to views were considered in Keller and Ullman [1984]. This paper pursues the Bancilhon-Spyratos hypothesis that view updates must preserve some “complement” of the view. A complement to a view is another view that together with the first is sufficient to reconstruct the database. We showed that in a simple model of databases, there is a unique minimal complement to a given view. Thus, these views have unique translations for their updates.

Hash Tables

Mairson [1983a] investigated the design of hash tables for efficient access to a preselected set of data, showing that under a wide variety of assumptions, the hash functions for such a purpose required a very long program to compute. This paper received the Machtey award as the best student paper at the 1983 IEEE Symposium on Foundations of Computer Science.

Database Logic

There have been several attempts to generalize relational calculus to a logical model and language that captures the generality of nonrelational database systems. However, Jacobs’ “database logic” allows noncomputable queries, and the “format model” does not capture indirection through pointers, as found in most nonrelational models. A logic that fixes these problems was described in Kuper and Vardi [1984]. It shows how high-level algebraic or logical languages can be designed for any of the popular models, not just the relational.

Knowledge-Bases and Databases

In work just begun as the grant expired, Ullman [1984] and Sagiv and Ullman [1984] explored database systems that have logical rules, in the form of Horn clauses, that interpret the data in a relational database. This arrangement is what AI folks call a “knowledge base.”

We developed some promising techniques for giving the system the ability to deduce the proper, or most efficient, way to deal with queries involving logical rules. The effect is that while languages like Prolog or MRS require the user to specify such strategies, systems
based on our ideas will be able to make their own decision. The situation is analogous to the way relational systems take from the user the responsibility for navigation that is present in network or hierarchical systems.

This work, although just beginning, has attracted considerable interest and research support. It is sad that AFOSR has elected not to pursue this direction.

Ph. D. Theses Support by Grant

Three Ph. D. students were supported by the grant. Their theses were: Korth [1981], Mairson [1983b], and Kuper [1984]. The first was actually awarded by Princeton Univ., since the student involved followed the PI when he moved to Stanford. The last of these has been written, but the degree will not be awarded until approximately June, 1985.

Bibliography of Grant Publications

Note: Many works supported by the grant were published in several forms, e.g., as a technical report, conference paper, and refereed journal paper. In such cases, only the final or most recent form of the publication is listed.


2/1/85