ADVANCES IN

COMPUTING RESEARCH

Guest Editor: PARIS KANELLAKIS

Series Editor: FRANCO P. PREPARATA

Volume 3 • 1986

THE THEORY OF DATABASES

ADVANCES IN COMPUTING RESEARCH

A Research Annual

THE THEORY OF DATABASES

Guest Editor: PARIS C. KANELLAKIS
Department of Computer Science
Brown University

Series Editor: FRANCO P. PREPARATA
Departments of Electrical Engineering and Computer Science
University of Illinois

VOLUME 3 • 1986

JAI PRESS INC.

Greenwich, Connecticut *London, England*

36 Sherwood Place
Greenwich, Connecticut 06836

JAI PRESS INC.
3 Henrietta Street
London WC2E 8LU
England

ISBN: 0-89232-611-5

Manufactured in the United States of America

ADVANCES IN COMPUTING RESEARCH

Volume 3 • 1986

THE THEORY OF DATABASES

To the memory of
WITOLD LIPSKI, JR.

On May 30, 1985, Witold Lipski, Jr. succumbed to an unforgiving illness in Nantes, France.

After earning his doctorate from the Polish Academy of Sciences in 1975, he joined the faculty of the Academy's Institute of Computer Science. At the time of his death he was a visiting professor at the Universite' de Paris-Sud, Orsay, France.

Witek was a productive and well-recognized researcher in various areas of theoretical computer science. His main field of interest was database theory, but his contributions span a wide range of topics, from logic to combinatorial algorithms and computational geometry. His scholarly work appeared in some of the most prestigious journals of the profession and in the proceedings of major conferences.

Besides being a respected scientist with a broad variety of cultural interests, Witek was a wonderful human being. For those who had the good fortune to meet him, his memory will be a permanent source of inspiration.

LIST OF CONTRIBUTORS

Walter A. Burkhard
Department of Electrical Engineering and Computer Science
University of California
San Diego

Stavros S. Cosmadakis
IBM Research Laboratories
Yorktown Heights, New York

Alessandro D'Atri
Department of Information and Systems
"La Sapienza" University
Rome, Italy

Ronald Fagin
IBM Research Laboratories
San Jose, California

Patrick C. Fischer
Department of Computer Science
Vanderbilt University

Gösta Grahne
Department of Computer Science
University of Helsinki
Helsinki, Finland

John Grant
Department of Computer Science
Towson State University

M. Gyssens
Department of Mathematics and Computer Science
University of Antwerp
Antwerp, Belgium

Paris C. Kanellakis
Department of Computer Science
Brown University

Gabriel M. Kuper
Department of Computer Science
Stanford University

Nancy A. Lynch
Laboratory for Computer Science
Massachusetts Institute of Technology

David Maier
Oregon Graduate Center
Beaverton, Oregon

F. Manfredi
CRAI
Rende, Italy

A. Mecchia
CRAI
Rende, Italy

Jack Minker
Department of Computer Science
University of Maryland
College Park, Maryland

Marina Moscarini
Institute of Systems Analysis and Information of the CNR
Rome, Italy

J. Paredaens
Department of Mathematics and Computer Science
University of Antwerp
Antwerp, Belgium

Kari-Jouko Räihä
Department of Mathematical Sciences
University of Tampere
Tampere, Finland

David Rozenshtein
Department of Computer Science
Rutgers—The State University of New Jersey
New Brunswick, New Jersey

D. Saccà
CRAI
Rende, Italy

Edward Sciore
Department of Computer Science
State University of New York at Stony Brook

Dale Skeen
IBM Research Laboratories
San Jose, California

Stan J. Thomas
Department of Computer Science
Vanderbilt University

Jeffrey D. Ullman
Department of Computer Science
Stanford University

Moshe Y. Vardi
IBM Research Laboratories
San Jose, California

David S. Warren
Department of Computer Science
State University of New York
at Stony Brook

David D. Wright
Prime Computer, Inc.
Framingham, Massachusetts

Mihalis Yannakakis
ATT Bell Laboratories
Murray Hill, New Jersey

EDITORS' FOREWORD

This is the third volume of the annual *Advances in Computing Research*, a series whose goal is the timely publication of archival articles in selected active areas of computer science research. Although the emphasis is on research of a theoretical nature, a strong relevance to currently significant practical applications is the guiding criterion for the choice of the theme of each issue. Thus, after one issue on computational geometry and another on VLSI theory, this volume is devoted to the theory of databases.

Each volume in this series can be likened to a "Special Issue" of a professional journal, with the added feature of a generally less terse expository style, as suggested and afforded by the book format. The operating procedure is similar to that of the most respected journals in the discipline: All the articles in each collection, once solicited through an editorial announcement, are subjected to the usual, rigorous peer-review process and selected on that basis.

The basic insight that "data should be treated as an integrated resource, independent of application programs," and the practical need for efficient manipulation of large amounts of structured information led, early on, to the development of database management as an important area of computer science research. The database approach to information management has had a major impact on software systems and computer science in general. In fact, it has provided one of the few paradigms of man–machine interaction which is both of very high level, akin to programming in logic, and computationally efficient. Database theory grew as the system theory corresponding to and often directly influencing a number of milestone database management system implementations. This branch of theoretical computer science has been a thriving area of activity during the past decade.

Database technology presents the theoretical community with challenging research questions, which can be classified into three broad categories: problems of *relational logic*, problems of *transaction processing*, and problems of *access methods*. The unity of research in these seemingly dissimilar areas has been provided by the database management systems themselves. These are large integrated software systems, which must address issues in all three areas, and which have been consistent test beds for the feasibility of many of the proposed theoretical solutions.

Of the three facets of database theory, relational logic is perhaps the one more closely identified with the field. The pioneering work, in the early 1970s, of E. F. Codd on the relational model of data offered the theoretical community an elegant, well-motivated, and appropriately restricted model of computation. In this framework a set of relations, or database, is the meaning of a set of nonlogical relational symbols or database schema. According to this philosophy, data is represented by finite relations, which are definable in first-order relational calculus—a special form of predicate calculus. An algebraization of this calculus, known as relational algebra, forms a convenient data manipulation facility. The duality between the declarative (calculus) and the procedural (algebra) approaches to database querying is at the heart of relational logic. Query languages—mostly of very high level by current programming language standards—have been extensively investigated, in terms of both expressive power and program optimization. A major issue, however, in this pure framework is the inability to constrain the possible databases and consequently express more knowledge about the entities and relationships represented. This deficiency has been remedied through the device of data dependencies—semantically meaningful and syntactically restricted sentences of predicate calculus added to the database schema. Studies of the decision problem and other computational properties of

data dependencies have been, primarily, motivated by questions of good database schema design, but have also contributed to basic research in mathematical logic. In addition, relational theory has been extended and enriched in order to address the problems of incomplete information and dynamic change (i.e., updates) in the database environment. A consequence of this development is that relational logic provides a standard of comparison for other data models. In a larger setting this part of database theory, which consists largely of techniques from logic applied to computer science problems, may be viewed as a concrete advance in understanding the foundations of programming in logic.

The emphasis of the two other facets of database theory is on algorithms and data structures. Transaction processing, despite its many connections to the theory of operating systems, has a unique database character. The main issue here is the preservation of data integrity and security in a parallel and sometimes unreliable environment of user programs or transactions. Database concurrency control, the safe and efficient scheduling of individual operations of concurrent transactions, is the best known problem in this area. A more general theory of reliable transactions, with strong ties to distributed computing, is the object of much current research.

The last branch in this threefold classification, the study of access methods, is the area closest to the theory of data structures. Its distinguishing feature is the preoccupation with secondary storage media and very large volumes of highly structured data. Algorithmic improvements on external searching and on the execution of algebraic queries are essential for the efficiency, which has made databases a model for other forms of high-level programming.

Today, logic programming implemented in a highly parallel and reliable fashion is recognized as one of the most promising directions in computing research. This is in many ways the natural evolution of the database approach to computation. It incorporates more powerful, recursive computation features in the relational model of data and has similar goals of computational efficiency and a robust programming environment. Database theory, which in a sense is coming of age, is the principal starting point for the identification of more general paradigms in this challenging broader context.

As usual, an editor is confronted with the complex task of organizing a collection of articles. Although a classification of the contributions is always somewhat arbitrary, the articles in this collection can be naturally grouped according to the aspects of database research outlined above. Particular emphasis is given to relational logic, a fact reflecting the vigor and technical maturity of the subject.

The first article in this collection proposes a radically new addition to

the foundations of database theory. Its starting point is the problem of updating a database. An update must satisfy certain integrity constraints and, if performed on a user view of the data, must be translated into a corresponding change of the entire database. "Updating Logical Databases," by R. Fagin, G. M. Kuper, J. D. Ullman, and M. Y. Vardi, reaches out of the limits of relational logic. New semantics for insertion and deletion of data are proposed. They are based on a representation in terms of sets of logical theories or "flocks." The broader logical framework and novel notions, such as "equivalence forever" (i.e., database equivalence that is preserved under updates), illustrate the new challenges of database theory.

The next four articles form a coherent unit on the application of hypergraph techniques to database schema design. Schemes whose attribute structure corresponds to an acyclic hypergraph occur frequently and have nice mathematical properties. "Characterizations for Acyclic Database Schemes," by G. Grahne and K. J. Räihä, provides alternative characterizations for acyclic schemes (of various degrees) in terms of data dependencies. The existing characterizations for α-acyclicity are complemented by new ones for β-, γ-, and Berge-acyclicity. The article "Recognition Algorithms and Design Methodologies for Acyclic Database Schemes," by A. D'Atri and M. Moscarini, extends the GYO algorithm, for testing α-acyclicity, to handle the other acyclicity degrees. This extended procedure recursively reduces the set of edges of a hypergraph by eliminating each edge that satisfies a suitable "pruning" predicate. It provides a simple, homogeneous, and efficient test of acyclicity degree and can, furthermore, be used as part of an incremental step-by-step design methodology. Decomposition of join dependencies into acyclic and cyclic parts is a generalization of the above tasks in database design. In "The Decomposition of Join Dependencies", M. Gyssens and J. Paredaens analyze the algebraic invariants of a particular decomposition method, the "Hinge Decomposition Algorithm". The dependencies commonly present in a database scheme consist of one (full) join dependency and a set of functional dependencies. Under this assumption various classes of schemes, such as the acyclic or the independent ones, enjoy a number of useful computational properties. New additions to the list of properties of these schemes are made in "Properties of Database Schemata with Functional Dependencies," by D. Saccà, F. Manfredi, and A. Mecchia. Their investigation also leads to new classes of schemes, the L-acyclic schemes for instance, whose relationship to predefined classes is described.

Functional and inclusion dependencies are used to express two important types of constraints on relations. Whereas a functional dependency (FD) restricts a relation to represent a functional relationship, an inclusion dependency (IND) serves to denote set containment. Two

articles in this collection are devoted to the expressive power of IND's and to the computational complexity of their combination with FD's. In "Comparing the Universal Instance and Relational Data Models," E. Sciore compares the power of the relational model with inclusion dependencies to another common way of expressing containments in a database scheme, known as the universal instance (UI) model. Assumptions implicit in the UI model are made explicit using IND's. Also, suitable syntactic restrictions are given for IND's so that the two models can be proved equivalent. "Functional and Inclusion Dependencies: A Graph Theoretic Approach," by S. S. Cosmadakis and P. C. Kanellakis, is a study of the implication problem for FD's and IND's. A new graph theoretic methodology is introduced and is used to demonstrate that a bounded and complete axiomatization cannot exist for FD's under pairwise consistency, i.e., when all possible typed IND's hold. New lower bounds are provided for the implication of FD's and typed, acyclic IND's Interestingly, acyclic IND's were introduced in the preceding article by E. Sciore.

The three articles which follow have a common underlying theme, the treatment of incomplete information. A weak instance for the database is any state of the world (relation over all possible attributes or universal relation) that is consistent with the known properties (the dependencies) and the facts (the tuples) recorded in the relations of the database. There may be true facts, however, which do not appear in the database. Since more than one weak instance may be the current state of the world or universal relation, it is reasonable to require the answer to a given query to consist of those facts which are true in all possible weak instances. These are the facts that can be deduced from the contents of the database and the dependencies. "Querying Weak Instances," by M. Yannakakis, examines the problem of translating a query on the universal relation into one involving the database relations, when the only dependency is the join dependency of the database schema. It is shown how to translate monotone queries. Furthermore, the size of the translation is related to the size of Boolean formulas computing NP-complete problems. This is suggestive of unavoidable worst-case exponential growth. D. Maier, D. Rozenshtein, and D. S. Warren, in their article "Window Functions," generalize the universal-relation/weak-instance framework. Window functions map databases, over a schema, into universal relations, over all attributes in this schema. All existing universal relation interfaces can be described in this framework. Two critical properties that any reasonable window function should satisfy are introduced and analyzed.

A logical treatment of incomplete information is attempted in "Answering Queries in Indefinite Databases and the Null Value Problem." J. Grant and J. Minker investigate answering queries for databases

that may contain indefinite data. Indefinite data is represented using disjunctions of tuples, and, unlike the weak instance case, only facts recorded in the database can be true—a closed world philosophy. Algorithms are developed to check candidate answers to queries and to find all "minimal" answers. Null values, a traditional way of describing incomplete information, can be investigated as a special case.

The last of the eleven articles on relational logic lays the groundwork for a database model which is tabular in nature and yet allows hierarchical or aggregated structures, as well as the usual flat tables on the relational model. In "Nested Relational Structures" S. J. Thomas and P. C. Fischer examine two algebraic operators NEST and UNNEST which restructure unnormalized relations by creating and removing embedded hierarchical structures. This modification of the relational model enhances its expressibility without the device of data dependencies.

The area of transaction processing is represented in this collection by two articles, both addressing the issue of reliability. In the first one, "Increasing Availability in Partitioned Database Systems," D. Skeen and D. D. Wright study strategies that allow a distributed database management system to continue functioning correctly, even after being divided into disconnected parts by a partition failure. The methods proposed and examined rely on dividing transactions into classes, according to the items accessed, and on the use of multiple versions of the data items. The correctness condition maintained is serializability of the sequence of transaction operations executed. The second article in this group "Concurrency Control for Resilient Nested Transactions," by N. A. Lynch, extends serializability theory to transactions with failures. Block-structured or nested transactions offer a higher degree of resiliency and have been implemented using a variant of two-phase locking (a popular concurrency control technique) known as Moss' locking algorithm. The new theory is illustrated through a rigorous proof of correctness of this, quite complex, distributed algorithm.

The subject of the last article of this volume, "Index Maintenance for Non-Uniform Distributions," by W. A. Burkhard, is a promising new access method. An adaptive data structure allows very efficient execution of insert, delete, update, and range query as on-line operations—a problem known as interpolation search. These operations are performed on records whose keys are distributed according to an unknown distribution function. The heart of the method is a table, approximating the unknown distribution, which is dynamically updated by each insert and delete operation. The scheme inherits all the advantages of address calculation strategies for interpolation search of uniformly distributed records.

This completes the synopsis of the articles in this collection. The variety and quality of the results attest to the continuing vitality of the discipline. We wish to thank all the authors for their contributions and the referees for their invaluable service in maintaining the high standards of this series. A special acknowledgement is due to R. Fagin for identifying in the preliminary extended abstracts presented at PODS '84 Conference a potential source of contributions to this volume. Indeed, nine of the articles included in this collection are the expanded and thoroughly revised versions of material first communicated at that meeting.

F. P. Preparata
Series Editor

P. C. Kanellakis
Guest Editor

CONTENTS