ADVANCES IN DATA BASE THEORY

Volume 2

Edited by

Hervé Gallaire

CGE—Laboratoire de Marcoussis
Marcoussis, France

Jack Minker

University of Maryland
College Park, Maryland

and

Jean Marie Nicolas

Centre d'Etudes et de Recherches de Toulouse
Toulouse, France

PLENUM PRESS · NEW YORK AND LONDON

Library of Congress Cataloging in Publication Data

Main entry under title:

Advances in data base theory.

"Based on the proceedings of the Workshop on Logical Bases for Data Bases, held December 14–17, 1982, at the Centre d'études et de recherches de l'Ecole nationale supérieure de l'aéronautique et de l'espace de Toulouse (CERT), in Toulouse, France. —T.p. verso.

Includes bibliographies and indexes.

1. Data base management. I. Gallaire, Hervé. II. Minker, Jack. III. Nicolas, Jean Marie. IV. Workshop on Formal Bases for Data Bases (1979 : Toulouse, France)

QA76.9.D3A347 001.64'2 81-116229

ISBN 0-306-41636-0 (v. 2)

Proceedings of the Workshop on Logical Bases for Data Bases, held December 14–17, 1982, at the Centre d'Etudes et de Recherches de l'Ecole Nationale Supérieure de l'Aéronautique et de l'Espace de Toulouse (CERT), in Toulouse, France

A Division of Plenum Publishing Corporation
233 Spring Street, New York, N.Y. 10013

Printed in the United States of America

FOREWORD

This is the third book devoted to theoretical issues in databases that we have edited. Each book has been the outgrowth of papers held at a workshop in Toulouse, France. The first workshop, held in 1977 focused primarily on the important topic of logic and databases. The book, *Logic and Databases* was the result of this effort. The diverse uses of logic for databases such as its use as a theoretical basis for databases, for deduction and for integrity constraints formulation and checking was described in the chapters of the book.

The interest generated by the first workshop led to the decision to conduct other workshops focused on theoretical issues in databases. In addition to logic and databases the types of papers were expanded to include other important theoretical issues such as dependency theory which, although it sometimes uses logic as a basis, does not fit with our intended meaning of logic and databases explored at the first workshop. Because of the broader coverage, and because we anticipated further workshops, the second book was entitled, *Advances in Database Theory - Volume 1*. The book "Logic and Databases" should be considered Volume 0 of this series.

The current book, *Advances in Database Theory - Volume 2*, is an outgrowth of a workshop held in Toulouse, France, December 14-17, 1982. As with the earlier workshops, the meetings were conducted at the Centre d'Etudes et de Recherches de l'Ecole Nationale Superieure de l'Aeronautique et de l'Espace de Toulouse (C.E.R.T.). We are pleased to acknowledge the financial support received from the Direction des Recherches Etudes et Techniques de la Delegation Generale pour l'Armement (D.R.E.T.), and from C.E.R.T. that made the workshop possible.

As was the case for its predecessors, the chapters of this book are based on substantially revised versions of papers presented at the workshop. Each chapter included in the book was reviewed by at least three experts in the field — both individuals who attended the workshop and others who did not attend the workshop. In addition, every paper was reviewed by at least one of the editors who

was responsible for recommending the paper for inclusion in the book. We are indebted to our referees for their thorough review and constructive comments made on the papers. Their comments served to substantially improve each paper.

This book, as well as the previous books, can be used as the basis of a graduate seminar in computer science. Students should have a first level course in database systems and some background in mathematical logic and algebra.

The book starts with an introductory section which summarizes achievements in each paper. Background material is not covered since it may be found partly in the introductions to the previous volumes and in books that have been published. Following this introduction, the chapters in the book are grouped into five sections devoted respectively to

(1) Database Schema Design: Cycles and Decomposition

(2) Integrity Constraints

(3) Incomplete Information

(4) Abstract Data Types for Formal Specifications and Views

(5) Query Language Theory

Our grateful appreciation goes to Constance Engle who typed the the entire book. We also wish to thank Brenda Mauldin, Susan McCandless and Deven McGraw for their assistance with the book and in the development of the subject and name indexes. Support for work on the book was also provided by the Air Force Office of Scientific Research under AFOSR 01-5-28068, and from the National Science Foundation under NSF Grant 01-5-23247.

H. Gallaire
J. Minker
J. M. Nicolas
October 1983

INTRODUCTION

The field of databases is still in a state where many practical questions need to be answered. Even in cases where alternative solutions have been developed few guidelines exist as to which solutions should be preferred and why they are to be preferred. The following questions are among those raised when defining database schemas and manipulating database information:

- How, and by what criteria, does one choose a type of schema?
- How does one specify a particular schema corresponding to an application?
- How does one express, and conveniently handle, database constraints?
- How does one deal with real-world properties such as the incompleteness of information?
- How does one express user queries in a simple manner, given a database schema?

Corresponding to each such practical question one can develop a theoretical framework in which a better understanding of the issues can be achieved. This volume of articles deals with the theoretical counterpart of some of the above questions. As such, it covers only a few of the many much needed investigations reported in the literature and devoted to theoretical aspects of databases. It is not surprising that with the many problems that need to be solved, alternative formalisms need to be introduced. Mathematical logic is one such formalism. It is certainly the one most used as it is applicable to many database problems, not only that of developing a deductive capability for a database. Other formalisms that have been used include algebra, and the theory of hypergraphs. Each of the five sections of the book, excluding this introductory section, addresses a specific topic. Each topic covers practical questions attacked either by mathematical logic, algebra or hypergraph theory. The five major topics covered in the book are:

1. Database Schema Design: Cycles and Decomposition.
2. Integrity Constraints.
3. Incomplete Information.
4. Abstract Data Types for Formal Specifications and Views.
5. Query Language Theory.

Below we provide an overview of the chapters in each section. The reader is assumed to have a background in the areas of research covered in the chapters. Background material can be found in standard texts or articles listed below.

- Logic:
  Introduction to Mathematical Logic by Mendelson [1964].
- Databases:
  Principles of Database Systems by Ullman [1980], or
  The Theory of Relational Databases by Maier [1983]
- Logic and Databases:
  Logic and Databases by Gallaire and Minker [1978] and
  Advances in Database Theory - Volume I, by Gallaire, Minker and Nicolas [1981] (in particular the introductory chapters to these books).
- Graphs and Hypergraphs:
  Graphs and Hypergraphs by Berge [1970].
- Algebra Theory:
  An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types, by Goguen, Thatcher and Wagner [1976].

Section 1. Database Schema Design: Cycles and Decomposition

The purpose of design theory for relational database schemas is to characterize schemas which present desirable properties with regard to data representation and data manipulation. Central to this theory are the notions of dependencies, normal forms, lossless decompositions, dependency preservation and that of schema acyclicity introduced only recently. As argued by its promoters this last notion is interesting because it characterizes schemas for which various problems can be solved by efficient algorithms and schemas particularly appropriate for query evaluation on the so-

called universal relation interfaces. An increasing number of researchers are now investigating this new notion of acyclicity and its various impacts; thus not surprisingly, four of the five chapters in this section are concerned, to various degrees, with cyclic/acyclic schemas.

The chapter by Biskup and Bruggemann is devoted to the development of a method that synthesizes an acyclic database schema in third normal form (3NF) which meets the lossless join and dependency preserving conditions. This method achieves acyclicity by extending a previously defined synthesis algorithm. It is based on properties of acyclicity which are exhibited in this chapter; some of them present an interest which goes beyond their use for defining the method.

The results presented by Ausiello, D'Atri and Moscarini should have significant implications on the efficiency of query evaluation on a universal relation interface. They study various minimality notions of attribute set covering by a set of relation schemas, and determine the computational complexity of finding such minimal coverings while relating it to the degree of acyclicity of the database schema.

Hanatani introduces the notion of "simplicity" as an extension to the concept of acyclicity for characterizing "superficial" cycles. A simple join dependency (JD) can be decomposed into two orthogonal parts: an acyclic JD which has the same multivalued dependency (MVD) structure as the initial JD and a set of embedded ("local") JDs. In other words, Hanatani provides an interesting characterization of relation schemas which imply the same MVDs as some acyclic relation schema.

Finally, the last two chapters in this section, respectively by Gyssens and Paredaens, and by De Bra and Paredaens, are devoted to relation decomposition. Gyssens and Paredaens present a methodology for decomposing cyclic (as well as acyclic) relation schemas. The proposed methodology is proven to decompose any decomposable relation and to produce a set of nonredundant relations. The kind of decomposition dealt with by De Bra and Paredaens, horizontal decomposition, does not relate to problems associated with acyclicity/cyclicity. Horizontal decomposition is introduced to handle "exceptions" to functional dependencies (FDs), which, in practice, are quite important. A clear theoretical treatment is given of "almost FDs", and appropriate normal forms are defined.

Section 2. Integrity Constraints

For almost ten years a number of papers have been published on integrity constraints. However, most papers were devoted to state (or static) constraints and to their classification according to various criteria. A number of papers have been concerned with finding efficient techniques to enforce integrity constraints on databases and an adequate formalism to express transition constraints. Two of the three chapters in this section deal with enforcement techniques, whereas the third is concerned with transition constraint formulation.

Henschen, McCune and Naqvi rely on a logical formulation of both integrity constraints and database transactions (updates, insertions, and deletions) to propose and justify a method based on theorem-proving techniques, for generating, at compile time, all specific tests. These tests are to be performed (and they can be performed in a much more efficient way) in place of constraint evaluation when actual database changes arise. As noted in the chapter, some open theoretical problems remain. Nevertheless this approach is quite convincingly argued and can be considered as a most promising one for efficient integrity enforcement.

Another interesting approach is that proposed by Paige. He shows how techniques of finite differencing, he developed elsewhere, can be applied to improve the enforcement of integrity constraints. The technique relies on maintaining some extra data, termed "differential" stored views, which can be used for reducing significantly the constraint evaluation cost. Of course, the maintenance cost of differential views should be low for the method to be beneficial. Paige characterizes a class of constraints for which this is the case.

Casanova and Furtado present a family of temporal languages to describe transition constraints. They prove various results about the decision problem for these languages, investigate their expressive power and compare their approach using temporal logic with an approach that uses a first-order logic formulation.

Section 3. Incomplete Information

Very few systems attempt to deal with incomplete information because, in the presence of incomplete information, one has to appeal to more elaborate question-answering and integrity maintenance techniques. One way to deal with this problem is to formulate the database in logical terms and then use the proof theoretic view of a database to formalize some form of nonmonotonic reasoning. Nonmonotonic reasoning does not guarantee the validity of previous

answers when more information is added. Bossu and Siegel describe a system of nonmonotonic reasoning based on the notion of ordering of interpretations and of minimal models. This form of reasoning handles formulas expressive enough for integrity rules, transition rules, and queries. The system can check, for any transaction made to a database, whether it is valid with respect to integrity and transition rules. The proof procedure they present is complete. The work is important in that it handles incomplete information correctly. However, this approach seems to need additional work to be computationally efficient in general.

In the same vein, namely the need to enlarge the type of information, queries and answers in a database system, Imielinski takes on an algebraic approach to answering queries in an attempt to avoid costly theorem proving techniques. He shows how extensions to relational algebra methods allows one to approximate answers in a database that contains incomplete data. He first defines an answer to a query when the database is a collection of arbitrary formulas. By representing formulas by tables (termed V-table), classical algebraic manipulations can be extended to such formulas. However, if a precise answer is needed he must resort to the logic proof theoretic view of databases. His method attempts to combine the best features of the relational algebra while resorting to theorem proving techniques for only a few cases.

Section 4. Abstract Data Types for Formal Specifications and Views

The abstract data types formalism has been used extensively to specify data structures and operations made on them. Since a database is a collection of data structures on which specific transactions are allowed, the formalism should be applicable to the specification of a database. Veloso and Furtado provide a specific multistep method to specify a database. The method proceeds from a high level specification to an executable one. At each level the algebraic style is preserved. The major idea is to specify a state of the database by the trace of transactions that yield the state. Equivalence of traces is formally defined and forms the basis of transformations applied at each step of the method.

A different use of abstract data type theory is made by Paolini and Zicari whose goal is to provide a framework for studying database views. Because views are abstractions of databases, algebra theory gives a formal expression of views and databases: a view can be obtained by a morphism applied to the database. Depending on mathematical properties of morphisms, views behave differently under user operations and thus view updates affect the underlying database differently. A classification of views is then obtained that relies on notions such as complete, total, strongly consistent views.

Section 5. Query Language Theory

No agreement has been reached on the choice of query languages with respect to such factors as expressive power, or user friendliness. Fundamental to possibly answering these questions is the need to study the equivalence of queries. Such a study should also help to solve query optimization problems. Restricting themselves to the attribute relational algebra, based on natural joins and relations with columns corresponding to attributes and to the propositional relational algebra, Imielinski and Lipski show that for any reasonably complex database, equivalence and finite equivalence of queries are undecidable problems.

References

1. Berge, C. [1970] *Graphs and Hypergraphs*, Dunod, Paris.

2. Gallaire, H., and Minker, J., Eds. [1978] *Logic and Databases*, Plenum Publishing Co., New York, N. Y.

3. Gallaire, H., Minker, J. and Nicolas, J.-M., Eds. [1981] *Advances in Database Theory - Volume I*, Plenum Publishing Co., New York, N. Y.

4. Goguen, J. G., Thatcher, C. W. and Wagner, E. G. [1976] "An Initial Algebra Approach to the Specification, Correctness, and Implementation of Abstract Data Types", In: *Current Trends in Programming Methodology, Volume 3, Data Structuring*, (R. Yeh, Ed.), Prentice-Hall, Englewood Cliffs, N. J.

5. Maier, D. [1983] *The Theory of Relational Databses*, Computer Science Press, Inc., Potomac, MD.

6. Mendelson, E. [1964] *Introduction to Mathematical Logic*, D. Van Nostrand, New York.

7. Ullman, J. D. [1980] *Principles of Database Systems*, Computer Science Press, Inc., Potomac, MD.

CONTENTS

DATABASE SCHEMA DESIGN: CYCLES AND DECOMPOSITION

TOWARDS DESIGNING ACYCLIC DATABASE SCHEMES

Joachim Biskup and Hans Hermann Brüggemann

Universität Dortmund

Dortmund, West Germany

ABSTRACT

We present a modification of the synthesizing method (Bernstein [1976]), Biskup et al [1979]) to produce a third normal form, dependency preserving, lossless join decomposition of a universal relation scheme which is additionally acyclic. Our method essentially uses the options of the original method of how to group functional dependencies with the identical left hand side. Furthermore we present a decomposition theorem for acyclic database schemes.

INTRODUCTION

For the relational model of databases several formal methods for supporting the design of database schemes have been presented (see, for instance, Chapter 5 of Ullman [1980]). By these methods we can assure that the designed database scheme has certain desirable properties. Some of these properties, such as third normal form, lossless join, or dependency preservation, are exactly defined in terms of a given set of semantic constraints (mainly functional or multivaled dependencies), whereas other properties, such as avoiding update anomalies or eliminating redundancy, are stated more intuitively. Recently the notion of acyclic database schemes has been proposed as a further desirable property of relational database schemes. See for instance Beeri et al. [1981], Beeri et al. [1983], Chase [1981], Fagin et al. [1982], Goodman and Shmueli [1983], Hull [1983], and Maier and Ullman [1982]. A recent survey on acylic database schemes is presented in Fagin [1983]. Properties of acyclic database schemes are studied in the proceedings from which this book is drawn, by Ausiello et al. [1982], Gyssens and Paredaens

[1982], and Hanatani [1982]. For instance, for an acyclic database scheme we can evaluate queries unambiguously with respect to a universal relation interface (queries that are expressed by means of attributes without mentioning relation names).

Currently we concern ourselves with studying the problem of how to use, respectively modify, the known formal design methods in order to obtain an acyclic database scheme whenever it is possible. In this chapter we propose an elaboration of the synthesizing method, Bernstein [1976], Biskup et al. [1979], and present some general properties of acyclic database schemes.

The synthesizing method is directed towards producing a third normal form, dependency preserving, and lossless join decomposition of a universal relation scheme that is essentially given by a set of functional dependencies. A functional dependency is a semantic constraint of the form $R \mapsto A$ where R is a set of attributes and A is an attribute. A relation (set of tuples) satisfies $R \mapsto A$ if any two tuples of r that agree on R also agree on A. The synthesizing method roughly proceeds as follows:

input: A set of functional dependencies F.

method:

1. [achieve third normal form]

   Eliminate redundancies from F.

2. [achieve lossless join]

   If there is no $R \mapsto A \in F$ such that R contains a key, then determine a key K.

3. [achieve dependency preservation]

   For $R \mapsto A_1 \in F, \ldots, R \mapsto A_k \in F$ form a relation scheme $R \cup \{A_1,\ldots,A_k\}$; any dependency of F must be used at least once.
   Depending on step 2 take K as an additional relation scheme.

At each step this method is not fully specified because we have several options:

- how to eliminate redundancy,
- which key to use,
- how to group functional dependencies with identical left hand sides.

By making good choices of the options we may affect the acyclicity property of the output database scheme. For the third option of grouping this can be demonstrated by the following example.