Foundations of Databases

Foundations of Databases

Serge Abiteboul
INRIA–Rocquencourt

Richard Hull
University of Southern California

Victor Vianu
University of California–San Diego

Addison-Wesley Publishing Company

Reading, Massachusetts • Menlo Park, California
New York • Don Mills, Ontario • Wokingham, England
Amsterdam • Bonn • Sydney • Singapore
Tokyo • Madrid • San Juan • Milan • Paris

Sponsoring Editor: Lynne Doran Cote
Associate Editor: Katherine Harutunian
Senior Production Editor: Helen M. Wythe
Cover Designer: Eileen R. Hoff
Manufacturing Coordinator: Evelyn M. Beaton
Cover Illustrator: Toni St. Regis
Production: Superscript Editorial Production Services (Ann Knight)
Composition: Windfall Software (Paul C. Anagnostopoulos, Marsha Finley, Jacqueline Scarlott), using ZzTEX
Copy Editor: Patricia M. Daly
Proofreader: Cecilia Thurlow

Library of Congress Cataloging-in-Publication Data
Abiteboul, S. (Serge)
Foundations of databases / Serge Abiteboul.
Richard Hull, Victor Vianu.
p. cm.
Includes bibliographical references and index.
ISBN 0-201-53771-0
1. Database management. I. Hull, Richard, 1953–. II. Vianu, Victor. III. Title.
QA76.9.D3A26 1995
005.74'01—dc20 94-19295
CIP

1 2 3 4 5 6 7 8 9 10–MA–98 97 96 95 94

Alice: *To whom are you going to dedicate the book?*

Vittorio: *How about our parents?*

Riccardo: *Yes, and our spouses . . .*

Sergio: *. . . and the kids . . .*

Alice: *Hold it, that's a dozen people already! Don't you guys have some common ancestor?!*

All: *You're right! We* also *dedicate it to Seymour Ginsburg.*

Preface

Database theory is a relative newcomer to the field of computer science. Early data management systems were based on techniques from several classical areas of computer science, ranging from hardware and operating systems to data structures and programming languages. In the early seventies, a leap of abstraction from file systems produced relational databases and its accompanying theory, with logic as the catalyst. We believe that database theory has matured—that it has emerged as an elegant and robust part of science with its own identity. As such, it embodies its own peculiar brand of wisdom that deserves to be communicated not just to insiders, but to the computer science community at large.

In a nutshell, a database management system is a software system that enables the creation, maintenance, and use of large amounts of data. In contrast with many programming applications, the logical data structure—the "database schema"—used to structure a given data set is usually much smaller than the volume of that set. Furthermore, the data is persistent, evolving over time and surviving multiple invocations of the database management software. To increase usability, concurrent access to the data is usually supported with specialized protocols that guarantee a form of noninterference between interleaved transactions. Importantly, modern database management systems embody a distinction between the *logical level* and the *physical level*. The logical level focuses on an abstract representation of the data, along with languages to create, query and modify it; the physical level focuses on the underlying implementation, including the physical layout used to store the data, the indexing and clustering schemes, and the concurrency and recovery protocols.

Database theory has developed primarily around the logical level of databases. (A notable exception is concurrency control, which is not addressed in this volume.) A core of fundamental material on the relational model has become well established. It consists primarily of three paradigms for query languages (algebraic, calculus-based, and deductive) and the theory of dependencies. The theory of query languages, including issues of expressiveness and complexity specific to databases, is well developed. The marriage between databases and logic programming produced deductive databases, with the main focus on the deductive query languages. Dependency theory focused initially on formalizing and applying the disparate integrity constraints that commonly arise in practice, and it went on to relate constraints with query optimization and to develop a unifying perspective for them.

As a field, database theory draws on several areas, including mathematical logic, complexity, and programming languages. But the database context brings with it different assumptions, perspectives, and emphases. Relations can be viewed as predicates in the sense of logic, and the relational calculus as a specialization of the first-order predicate calculus. However, the database area emphasizes finite structures and has developed the notions of "domain independence" and "safety" to capture intuitive properties related to this finitude. The questions and techniques in dependency theory borrow heavily from logic, with a focus on practically motivated, relatively weak classes of sentences. Query languages provide an interesting contrast with conventional, imperative programming languages. Query languages typically embody a set-at-a-time focus as opposed to an object-at-a-time focus. Also, they are largely declarative in nature, and failing that, are more applicative than imperative. Because of the emphasis on tractability in the face of large volumes of data, there is considerable interest in query languages that do not have full computational power, which gives rise to a rich interplay between query languages and complexity theory. Specialized notions of complexity have arisen, stemming from the practical reality of large volumes of data and the theoretical interest in different query languages. Also, the important notion of "genericity," which captures a form of abstraction stemming from the separation of the logical and physical levels, has led to new perspectives on complexity theory, involving formalisms that circumvent the ordering of input data implicit in traditional Turing machines.

Exciting new research directions have continued to emerge in database theory, stemming primarily from unanswered questions about query languages and from an interest in expanding beyond the limitations of the relational model. Current research includes investigations motivated by connections with object-orientation, artificial intelligence, and graphics interfaces. And as the database field matures, it, in turn, influences adjacent areas in computer science, notably finite model theory, programming languages, and logic programming.

A Note on Style

This book deals with the theory that has developed around the logical level of databases. It has two main objectives: to provide a focused presentation of the core material and to present the essence of the advanced material in a unified framework. Some of the advanced material has never before been presented in book form. The presentation style is quite rigorous, in that precise definitions and statements of results are provided. However, our overriding concern was to make things simple to the reader, to get across the intuition and elegance of the concepts and proofs, rather than adhere to very strict criteria of rigor. Numerous examples, figures, and exercises should help clarify the development. Some of the proofs emphasize intuition and leave out much of the detail; we called such a proof a "crux." In this way we have tried to achieve a balance between formalism and intuition. As we went along, a two-tier style emerged, with a tendency towards more rigor in the exposition of the core material and more intuition in the presentation of advanced and tangential topics.

The book is aimed at an eclectic audience. Most broadly, it should be a useful resource for any computer scientist or mathematician who wishes to find out what database theory is about. Database researchers and practitioners should find it useful as a reference to both classical material and to advanced topics, otherwise scattered in sometimes hard-to-read papers. As a textbook, it is aimed at graduate students and seniors who would use the book as the main text in a database theory course or as complementary material in a database systems course. The book is fairly self-contained. Some needed background is provided concisely in the preliminaries, and the reader is told where more can be found.

We have attempted to make life easier for the database aficionado by adapting the material consistently to the database framework. This saves the reader the work of translating to the database framework, a task which is at best distracting and at worst tedious and confusing. Perhaps the most substantial difference from the conventional presentation arises in the case of logic programming, where the deductive database point of view has dramatic impact. Mostly, things become much simpler because there are no function symbols. However, for some questions, such as expressive power and complexity, the conventional approach is simply inapplicable. The book also maintains a strong focus on database theory issues—tangential material from adjacent areas has been kept to an absolute minimum.

Results are attributed to their sources in bibliographical notes, included at the end of each chapter.

Organization of This Book

The outline of this book is as follows. Part A contains preliminaries, and a brief introduction to databases and the relational model.

The basic material on relational query languages is developed in Part B. Throughout the entire presentation of query languages, we develop in parallel the three query language paradigms: algebraic, calculus-based, and deductive. The presentation starts with conjunctive queries, which lie at the core of virtually all query languages, both formal and commercial, and have particularly nice properties that make them the prototypical well-behaved class of queries. We use conjunctive queries as a vehicle to introduce many of the main issues in query languages, which will recur throughout the book. More powerful languages are then obtained by gradually enriching the conjunctive queries by adding negation, recursion, and finally negation and recursion combined. At each stage, we examine the basic properties of the newly obtained classes of queries and contrast them with queries considered previously. Adding union and negation yields three equivalent languages: relational calculus (first-order logic without function symbols), relational algebra, and non-recursive datalog$^\neg$.

The languages with recursion are studied in Part D. Adding recursion to the conjunctive queries yields datalog. Recursion in combination with negation produces datalog$^\neg$ (with the various associated semantics for negation), and other languages, all the way to the *fixpoint* and *while* queries. Included here is a presentation of the logic programming paradigm in the framework of databases, with implementation techniques, including top-down, bottom-up, and heuristics for optimization.

Dependency theory is presented in Part C. The emphasis is on functional, join, and inclusion dependencies, which are widely encountered in practice. We use these classes of dependencies to illustrate the main issues. A larger perspective on more general dependencies is also provided. Uses of dependencies in query optimization and schema design are presented.

More advanced material on the expressiveness and complexity of query languages is presented in Part E. The goal is to highlight the aspects of computability and complexity specific to databases. This includes complexity measures for queries and results on the connection between languages and complexity classes. In particular, the notion of query is formalized carefully, and languages that express all queries are exhibited. Some advanced techniques like 0-1 laws and games, are presented in an informal and self-contained way, with the aim of making them easily accessible to readers with a limited background in logic.

Finally, several advanced topics are briefly surveyed in Part F. They include incomplete information, complex objects and object-oriented databases, and dynamic aspects. These share the characteristic that they are less settled, and possibly more controversial than the other topics covered. Our aim is to identify the main concepts and issues in each of these areas, that are likely to generate a substantial portion of the database theory research in the coming years. The style is more informal, in the manner of a survey, with most proofs omitted and pointers provided to current research.

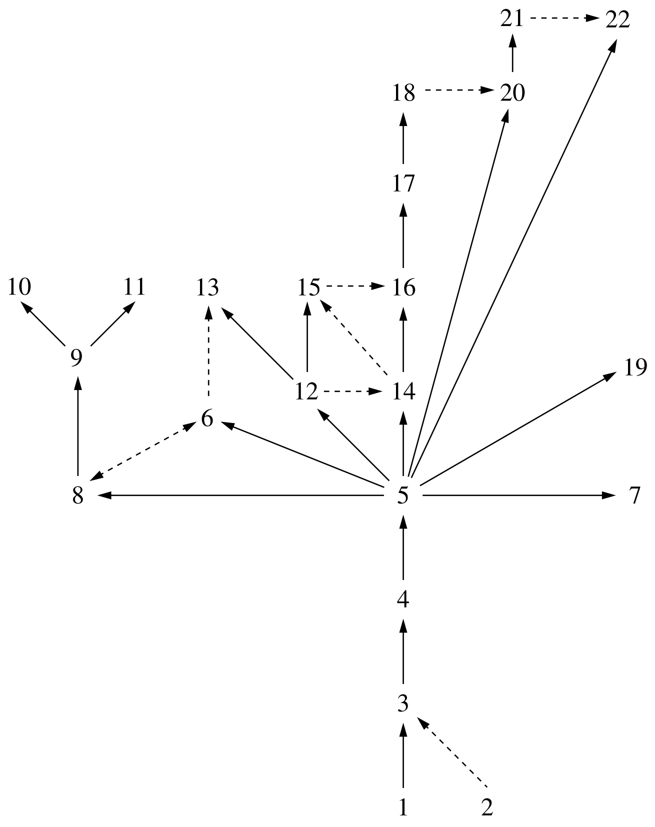
Teaching from This Book

To teach from this book, several paths can be followed. The diagram on page xi indicates dependencies among chapters. Solid arrows indicate prerequisites. Dashed arrows indicate preferred, but not mandatory, order among chapters. As a sample, here are some possible menus:

- The Classic: a basic database theory course covering the classical material would center around Parts B and C. Chapter 10 and parts of Chapter 9 of Part C are somewhat more advanced and could be skipped. If time allows, some of Chapter 12 and a selection of Part F might be covered.
- Feast of Query Languages: a course on the theory of query languages would start with a quick review of the basic material on classical languages (Part B), and continue with Parts D and E. If time allows, some material on languages for complex objects and object-oriented databases (Part F) could be covered.
- Gourmet Sampling of Database Theory: a course for people with theoretical appetites that emphasizes the specificity of database theory. Logicians wishing to explore the connection between finite-model theory and databases will be interested in Parts C and E. Those interested in descriptive complexity will find Part E closest to their hearts. Researchers in logic programming will prefer Part D, particularly Chapters 12, 13, and 15. People with a background in theoretical artificial intelligence will find Parts D and F of particular interest. Rule-based systems are related to Chapter 14 (see also parts of Chapter 22). Programming language people will be interested in much of the material on query languages, including Chapters 20 and 21 in Part F.
- Fast-Food Database Theory: a course for applied database students that is meant

to provide informal exposure to some of the basic results of database theory. This would include assorted topics, with an informal presentation emphasizing the examples, results, and intuition provided in the text, rather than the proofs. A possible syllabus would include Part B; parts of Chapters 8, 9, and 11 in Part C; Chapter 12 and parts of Chapter 15 in Part D; and selected chapters of Part F.

Numerous exercises have been included, and they are essentially of three categories. Routine exercises (unmarked) require easy manipulation of the basic concepts and results. Harder exercises are marked with a (★). Another category of exercises is meant to complement the material in the text and often contains results from related research articles. These exercises are usually on the hard side, and some may constitute term projects. These exercises are marked with a (♠).



Acknowledgments

We are grateful to several people for their many comments and fruitful discussions on the material of this book: Dennis Allard, Nicole Bidoit, Catriel Beeri, Luca Cabibbo, Sophie Cluet, Claude Delobel, Guozhu Dong, Sophie Gamerman, Françoise Gire, Gösta

Grahne, Erich Graedel, Stéphane Grumbach, Gerd Hillebrand, Dean Jacobs, Paris Kanellakis, Phokion Kolaitis, Leonid Libkin, Tova Milo, Jan Paredaens, Raghu Ramakrishnan, Michel Scholl, Jianwen Su, Jan Van den Bussche, Moshe Y. Vardi, and Emmanuel Waller. We are especially thankful to the following people, who served as formal reviewers of part or all of the manuscript: Luca Cabibbo, Jan Chomicki, Jeffrey F. Naughton, Daniel J. Rosenkrantz, Jan Paredaens, Raghu Ramakrishnan, Jianwen Su, Jan Van den Bussche, Dirk Van Gucht, Moshe Y. Vardi, and Marianne Winslett.

We also wish to thank our students from ENS–Paris, ENST–Paris, UCSD, USC, and CU-Boulder who suffered through early versions of the book and tested exercises. In particular, we thank Tim Bailey, Ti-Pin (Ben) Chang, Martha Escobar-Molano, Sergio Lifschitz, Alvaro Monge, Huda Omran, and Gang Zhou.

We are very grateful for the extensive support and assistance we received from the staff at Addison-Wesley, including notably Lynne Doran Cote, Katherine Harutunian, and Peter Shepard on the editorial side, and Helen Wythe on the production side; and the staff of Superscript Editorial Production Services. We also thank Eileen Hoff and Toni St. Regis for helping to develop and executing the cover illustration.

We would also like to acknowledge the continuing support of the NSF and the ESPRIT Program. Finally, the Verso Research Project at INRIA–Rocquencourt provided a warm and stimulating environment for much of our collaborative work.

SA, RH, VV
Paris