APPLIED DATABASE LOGIC

Vol. I
Fundamental Database Issues

Barry E. Jacobs

APPLIED DATABASE LOGIC I: Fundamental Database Issues

Barry E. Jacobs
Senior Research Computer Scientist
National Space Science Data Center
Goddard Space Flight Center
National Aeronautics and Space Administration
Greenbelt, Maryland 20771

PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging in Publication Data
Jacobs, Barry E., 1947–
Applied database logic I.

Includes bibliographies and index.
1. Data base management. I. Title. II. Title:
Applied database logic 1. III. Title: Applied
database logic one.
QA76.9.D3J33 1985 001.64 85–3499
ISBN 0–13–040205–2

Editorial/production supervision and
interior design: Fred Dahl
Manufacturing buyer: Gordon Osbourne

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-040205-2 01

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Whitehall Books Limited, *Wellington, New Zealand*

PREFACE

Data can be represented in a number of ways in a database. For example, there are the three major methodologies: relational, hierarchical, and network. Further, even for a particular methodology, there are different implementations from which to choose. For instance, in the relational approach there are Oracle, Ingres, dBase II, and so on. In the network case, although CODASYL is intended to be a standard, there are many variations among existing CODASYL systems.

This abundance has resulted in two significant problems. First, because of the diversity of approaches, a database professional often finds that reading the literature requires tedious translation between terminology with which the professional is familiar and that which appears in, say, a journal article. Second, database users wanting to access data in a database type other than their own have to learn new rules and operations for manipulating the data. Consequently, this has created an "it's too much trouble to do" attitude. As a result, a tremendous proliferation and repetition of database research papers and databases has developed.

Database logic, the focal point of this book, is a proposed remedy for this situation. In very simple terms, database logic can be thought of as a uniform layer that can be placed on top of heterogeneous databases (i.e., relational, hierarchical, and network). Consequently, it can provide a uniform paradigm for communicating results of database research. In addition, database logic can be used as a way of looking at heterogeneous databases so that user access is facilitated.

More precisely, database logic is to the heterogeneous cases what first-order logic is to the relational approach. In this book, which presumes no previous background in logic, we show how database logic can be used to deal with many important database issues. These include

1. External-to-conceptual mapping
2. View update
3. View integration
4. Database conversion and query processing
5. Automatic program conversion
6. External axiomatization

These issues are all seen to be related using the notion of "interpretation."

Many of the ideas to be presented have come from our experience on the Distributed Access View Integrated Database (DAVID) project funded by National Aeronautics and Space Administration (NASA). The objective of this project is the development of an easy-to-use system with which people in the federal government can uniformly access distributed heterogeneous databases. Basically, DAVID is a database management system which is built on top of already existing database and file management systems. The front end of the DAVID system is database logic.

This book is unique in several ways. First, it is the first text that contains a treatment of all the above-mentioned issues. Second, it presents a unified approach, namely, interpretations, for dealing with all these issues. Third, many of the results in this book are new and have not yet appeared in the literature. Fourth, it presents a complete treatise of the recently developed tool of database logic. Fifth, it provides a vehicle through which computer science students can learn mathematical logic and discover how it can be applied to important database issues.

This book is the first in a series. The main theme throughout is the demonstration of how database logic may be used as a uniform tool for addressing many important database issues. This book deals with the fundamental issues mentioned above and shows how the notion of interpretation can tie them all together. The second book will deal with query processing over a set of heterogeneous distributed databases. The third book will deal with expert systems that are built on top of databases. Other books will deal with machine learning, transaction processing, database machine architecture, and theoretical issues concerning database logic. Our *modus operandi* is simple: We generalize database technology from the relational to the heterogeneous case using database logic as the framework.

The development of this book and the material within would not have been possible without the help of a number of people. Alan R. Aronson, Dehe Cao, Upen S. Chakravarthy, John Grant, Cindy Walczak, and Jack Welch have all contributed, in many ways, to the material that is presented in this book. Milt Halem and Paul H. Smith of NASA have provided the support, both technical and moral, for the DAVID project which resulted in this endeavor. Michael Anshel, Phillip Bernstein, John Berbert, Marco Casanova, Seymour Ginsburg, Keith Harrow, Stephen Hegner, David

Hsiao, Jack Minker, Ray Reiter, Ira Sack, Ed Sibley, and Moshe Vardi have provided direction and reassurance.

There are a number of individuals who helped to test and proofread the manuscript. These are Joseph Aulino, Kenneth Baum, Nancy Broderick, Isadore Brodsky, Edmund D'souza, Jennifer Elgot, Hsin Fang, Liang Fang, Elizabeth Fineman, James Finucane, Hsiao-Fang Hu, Kyu-Hyun Hwang, Adel Gharib, Diana Gordon, Kim Haynes, George Kertz, Thomas Jacobs, Susan Mittelman, David Moraff, Karen Roberts, Gompa Raghuramaih, Michael Shapiro, Duc Tran, John Waclawsky, and Karen Weiss.

I wish to also express my deepest thanks to a number of people at Prentice-Hall who helped me through various stages of this book. These are Lynn Frankel, Rhoda Haas, Barb Klink, Leon Liguori, Nancy Milnamow, and Gordon Osbourne. Special thanks also to Ira Shore of PPI Inc. who gave me a course in computer typesetting; to Barbara Palumbo of Prentice-Hall, who taught me so much about book production; and Fred Dahl, who coordinated the editing, production, and design of this book.

I am tremendously grateful to Karl Karlstrom at Prentice-Hall, one of the true giants in the computer science publishing industry, for his unwavering support of this project. I also thank Raymond Yeh at the University of Maryland for his help in initiating the publication of this series. Most of all, I wish to express my gratitude to my wife, Susan, who walked with me through every step on the oftentimes rocky and frustrating road to this point. I love her dearly for it.

BARRY E. JACOBS

HOW TO READ THIS BOOK

In this section we provide the reader with helpful suggestions for reading this text. The way in which one goes through the material will depend on one's background and the topics of interest.

1. The Introduction should be read in its entirety. Its purpose is to provide the reader with some preliminary background as well as an overall view of the material in the book. As in all chapters in the book, doing the exercises at the end of each section is strongly encouraged.

2. Chapters 1 and 2 should be read in their entirety. They form the foundation for the rest of the book. Unfortunately, they will be the hardest parts to get through. This is because the notation and concepts will probably be foreign to most readers. The reader should spend more time on understanding the examples than in trying to figure out the nuances of the definitions. After completing the first two chapters, the reader will be able to proceed faster through the rest of the book.

Chapter 1 shows how database logic can be used to represent heterogeneous databases. In Section 1.1 we cover the special case in which first-order logic represents relational databases. This section is particularly useful for readers without any background in logic. In Section 1.2 the notions of Section 1.1 are generalized to the heterogeneous case in database logic. In Section 1.3 the ideas of the first two sections are illustrated by defining a data manipulation language based on database logic.

After finishing Chapter 1, the reader is not expected to be a proficient "logician." Instead, it is expected that the reader will be comfortable with database logic view definitions as exemplified by Figures 1.3 and 1.8. There are formal definitions of database logic in Chapter 3. These should be looked at while reading Chapter 1 only if the reader feels uncomfortable with the presentation.

Chapter 2 shows how the notion of "interpretation" can be used as a vehicle to relate two different databases. In Section 2.1 we cover the special case in which both databases are relational. In Section 2.2 we generalize the material of Section 2.1 to the heterogeneous case. In Section 2.3 we show how programs can be converted from one database to another.

After finishing Chapter 2, the reader should be comfortable with the interpretation definition as exemplified by Figures 2.3 and 2.11. Also, it is expected that the reader will be familiar with the two mappings produced by the interpretation and the "Flipper Theorem." Again, the reader should consult the formal material in Chapter 3 only if uncomfortable with the presentation in Chapter 2.

3. Chapter 3 contains the formal definitions of database logic. Section 3.1 presents a discussion of how the important issues mentioned in the Introduction can be framed using the concepts from Chapters 1 and 2. This section should be read in its entirety. Sections 3.2 and 3.3 contain the formal material. The reader can skim through them and move on in the book.

4. The reader can now choose to read any of the remaining chapters in the book. Chapter 4 should be read before Chapter 5; otherwise, the reader can jump around as he or she chooses. There are cases in which material occurring in one chapter is referred to in other chapters. For example, logical optimization of Chapter 8 is mentioned in Chapter 7. However, these situations do not occur too often, and when they do, the material can still be pretty well digested.

5. The general mode of presentation in this book is that issues are first covered for the relational case and then generalized to the heterogeneous case. If one is concerned only with the relational case, then one can study only those chapters and sections pertaining to it. In all cases, proofs are provided for the heterogeneous case. However, a relational reader can still follow the arguments simply by pretending that only the relational case is dealt with.

B.E.J.

CONTENTS

INTRODUCTION

0.1 DATABASES

Roughly speaking, a *database* is a collection of data and operations that represents some aspects of the real world. For example, a university database can be regarded as representing parts of a university environment. Hence, one may regard a database as a "model" of that part of reality in which one is interested. Consequently, when we use a database, questions about the real world are reduced to questions about the model.

There are three major types of databases: relational, hierarchical, and network. The differences between them are based on the way in which data in the database appear to the user. Illustrations of the three types are given below. A *database management system* is a large-scale computer program that is responsible for supporting many databases. Depending on the type of databases it maintains, a database management system is called relational, hierarchical, or network.

Relational Databases

Relational databases are databases in which data appear to the user in one or more separate tables. These tables, called *relations,* have rows that are called *records.* For example, Figure 0.1 contains an instance of a relational database. The database represents student transcript information at a university. The relation TRANSCRIPT has records that indicate the student id#, student name, semester, year, department, course#, section#, course title, grade, and credits. The relation STUDENT has records which have student id#, student name, and date of admission.

TRANSCRIPT									
STUDID	STUDNAME	SEMESTER	YEAR	DEPT	COURSE #	SEC #	TITLE	GRADE	CREDITS
123456789	SMITH ROBERT J.	FALL	1979	COMPSCI	103	001	INTRO	B	3
123456789	SMITH ROBERT J.	FALL	1979	MATH	103	001	CALC	B	3
123456789	SMITH ROBERT J.	FALL	1979	ART	101	001	INTRO	B	3
123456789	SMITH ROBERT J.	FALL	1979	HIST	102	001	AM HIST	C	3
123456789	SMITH ROBERT J.	SPRING	1980	COMPSCI	220	001	DISCR STRUCT	B	3
123456789	SMITH ROBERT J.	SPRING	1980	MATH	104	001	CALC	B	4
123456789	SMITH ROBERT J.	SPRING	1980	ECON	101	001	INTRO	C	3
123456789	SMITH ROBERT J.	SPRING	1980	ENG	105	001	COMPOS	A	3
361258823	JONES MARY T.	SPRING	1980	MATH	104	001	CALC	A	4
361258823	JONES MARY T.	SPRING	1980	ECON	101	001	INTRO	C	3
361258823	JONES MARY T.	SPRING	1980	PHYSICS	102	001	INTRO	D	4
361258823	JONES MARY T.	SPRING	1980	ENG	105	001	COMPOS	I	3

STUDENT		
ID#	NAMES	DATEADM
123456789	SMITH ROBERT J.	SEPT 2 1979
361258823	JONES MARY T.	FEB 1 1980

Figure 0.1 Relational database.

There are also semantic properties attached to a relational database, called *integrity constraints.* The most common type of integrity constraint is the *dependency constraint.* It is defined as a collection of fields in a record that uniquely determines other fields in a record. For example, in the TRANSCRIPT relation, STUDID determines STUDNAME, and DEPT and COURSE# determine TITLE and CREDITS. In STUDENT, ID# determines NAME and DATEADM.

When a set of fields determines the rest of the fields in the row, that set is called a *key.* For example, ID# is a key for STUDENT. Also, the reader can check that STUDID, SEMESTER, YEAR, DEPT, and COURSE# is a key for TRANSCRIPT.