Principles and Practice of Database Systems

S.M. Deen

Principles and Practice of Database Systems

S.M. Deen

Department of Computing Science
University of Aberdeen

M
MACMILLAN

First published 1985

Published by
Higher and Further Education Division
MACMILLAN PUBLISHERS LTD
Houndmills, Basingstoke, Hampshire RG21 2XS
and London
Companies and representatives
throughout the world

Printed in Great Britain by
The Camelot Press Ltd.
Southampton

British Library Cataloguing in Publication Data
Deen, S. M.
Principles and practice of database systems.--
(Macmillan computer science series)
1. Data base management 2. File organization
(Computer science)
I. Title
001.64'42 QA76.9.D3

ISBN 0-333-37099-6
ISBN 0-333-37100-3 Pbk

Preface

This book was originally intended as a new edition of my earlier book *Fundamentals of Data Base Systems*, published several years ago. But, in the intervening period, so much has changed in the field of databases that I thought it better to write an entirely new book – with a new framework to incorporate the change. However, like its predecessor, this book also draws heavily from my undergraduate course at Aberdeen, which has now been given for over ten years. A major source of contribution to my course, and more importantly to the contents of this book, has been the deliberations of the *British Database Teachers Conferences*, which I helped to organise in the late seventies and early eighties, with a view to improving the standard of database teaching in the U.K. I have also taken into consideration the comments and suggestions received from the readers of my earlier book. An area to which I have paid special attention is the need of potential research students for an advanced treatment of *some* database issues at the undergraduate level, as part of a general background. This is an open-ended area with considerable scope for disagreements as to what should or should not be included. In this I have been influenced by my involvements in a number of international conferences on database research, and by my experience as a supervisor of research students. Thus the book is intended as a general text on databases for undergraduate students and other interested readers, but has been extended to include some advanced topics for those who need them.

The book deals primarily with the general principles of databases, using the relational and network models as the main vehicle for the realisation of these principles. Its basic objective is to impart a deeper understanding of the issues and problems, rather than to produce database programmers. To reinforce what is learnt, exercises with selected answers are provided at the end of each chapter (except for chapter 1). The references cited can be looked up for further studies.

The book is divided into five parts, with Part I (chapters 1–3) presenting a prerequisite for the appreciation of the main database issues. Part II (chapters 4 and 5) covers the issues and problems, and thus constitutes the kernel of the book. Part III is devoted to the relational and network models and is spread over six chapters (6–11). A rationale of this part could be helpful.

I have presented the first three normal forms and the original relational languages (algebra, calculus, Alpha) as part of a basic (or classical) relational model in chapter 6. Needless to say, a sound knowledge of a relational algebra is essential for a proper understanding of the relational model, and Codd's calculus and Alpha are the principal sources of all subsequent calculus based languages. QUEL is included in chapter 6 as the closest implementation of Alpha. Admittedly SQL is the leading relational language, but I found it more convenient to present it in chapter 8 with QBE, rather than packing it in an already large chapter 6. Chapter 7 on higher normal forms stands on its own.

The Codasyl model is described in chapter 9 in fair detail. In keeping with the objective of this book, I have stressed the semantics more than the syntaxes – syntaxes are given mainly for a more concrete understanding of the concepts involved. An overview of the storage schema in chapter 10 outlines a possible storage model for the Codasyl schema, and also shows where the storage dependent clauses of the earlier schema models have been moved to. The Fortran subschema facility in chapter 10 illustrates the mechanism of supporting a different host language on the Codasyl model. Since the implemented Codasyl packages are based on the earlier models, the section on the past changes provides a link between the latest version and the one that the user may still have in his computer centre.

Obviously the ANSI network model presented in chapter 11 will eventually replace the Codasyl model. Additionally, it offers some new ideas on database/host language interfaces. The same chapter also carries a comparison between the relational and the network models, which attempts to fulfil a need recognised by many teachers.

Part IV (chapters 12 and 13) describes the current state of the art. Chapter 12 reviews user issues, while chapter 13 discusses some special products, including SYSTEM R which demonstrates many of the pioneering implementation techniques of the relational model. The other special products described there represent the earlier data models and hence their coverage highlights their modelling powers, rather than their supporting facilities. Since the hierarchical model (IMS) is regarded by many experts as the third major data model (after the relational and network models), I have described it more fully and also compared its modelling capacity with that of the network model.

The final Part V (chapters 14 and 15) attempts to provide a glimpse of the future, and includes such topics as I thought useful and exciting.

Department of Computing Science
University of Aberdeen

S. M. Deen

Acknowledgements

I wish to thank a number of people who assisted me during the writing of this book. To Ray Carrick, John Edgar, Don Kennedy and Malcolm Taylor of my PRECI research group at Aberdeen, for going through the manuscripts and for suggesting corrections. To David Bell (University of Ulster), Bill Samson (Dundee College of Technology), Stan Blethyn (Bristol Polytechnic) and the publisher's referee for reviewing the material and for their valuable comments. Stan Blethyn also supplied examination questions, some of which are included in the exercises. To Frank Manola (Computer Corporation of America, U.S.A.) for keeping me informed on the latest developments on the ANSI data models.

I am also thankful to Muriel Jones of IBM, Phillip Robling of Cincom and Geoff Stacey of Edinburgh University for reviewing the coverages on IMS, TOTAL and Codasyl DSDL respectively. My special thanks are to my secretary, Sheila Tomlins, for her painstaking typing and retypings of the manuscript. She was a tremendous help.

Finally, I take this opportunity also to express my indebtedness to my family for the support I received. To my wife Rudaina for her patience and encouragement and to two young men, Rami (12) and Alvin (10), for leaving their dad alone even when their computer programs needed debugging.

A Note to Tutors and Readers

As pointed out in the Preface, this book is aimed at undergraduate students and other interested readers such as programmers, systems designers and database administrators. The reader is assumed to possess some knowledge of computers and of Cobol as a general background, within which each topic in the book is developed – starting from a basic level and gradually progressing towards deeper issues. The contents of the book can be logically divided into two parts, basic and advanced. The chapters covering the basic material are

chapter 1
chapter 2
chapter 3
chapter 4
chapter 5 (except section 5.7 which is advanced)
chapter 6
chapter 9
chapter 12

(advanced chapters are marked with an asterisk * in the Contents list).

It should be possible to understand the basic material, by-passing the advanced parts, without any serious loss of continuity. Within each chapter there are issues which are treated in some depth, but a reader not interested in these issues should be able to skip over them, again without any significant loss of understanding of the rest of the material. Every basic chapter is largely self-contained; most of chapters 6 and 9 (the relational and the Codasyl models) can be understood at a preliminary level if – apart from chapters 1 and 2 – only sections 4.1, 4.2 and 4.3 are read beforehand. However, a more thorough understanding of relevant database issues requires the study of both chapters 4 and 5 in full.

For an introductory course on the relational model it should not be necessary to cover functional dependencies and determinants, since both the second and third normal forms have been given alternative definitions. Likewise, quadratic join, outer join and QUEL can also be skipped if so desired. The students can be asked to read relational calculus and DSL Alpha for themselves.

If a briefer coverage is intended for the Codasyl model, then exclude the CALL, CHECK and SOURCE/RESULT clauses, the STRUCTURAL constraint clause and the multimember set types. Also restrict the set order criteria to the DEFAULT option and set selection criteria to the current of set types. Correspondingly, the Alternate clause, the set selection criteria and the subschema working storage section can be dropped from the subschema. Chapter 12 is written mainly for beginners, to give them some notion of the database environments. It is suitable for self-reading by the students.

For advanced studies, chapter 3 should be covered along with other chapters as needed. Much of chapter 10, and the ANSI network and relational models in chapter 11, can be given to the advanced students to read for themselves, with a view to carrying out some of the exercises at the end of those chapters.

Many of the exercises in the book are drawn from Honours degree examination questions. Readers' attention is drawn particularly to exercise 6.9 of chapter 6, which includes a *cyclic* query, exercise 8.8 of chapter 8, which shows a limitation of the relational model, and exercise 9.12 of chapter 9, which illustrates the use of currency indicators. There are also three special case studies presented in exercises 8.7, 8.8 and 8.9 of chapter 8 for relational solutions, and in 9.9, 9.10 and 9.11 of chapter 9 for Codasyl solutions. These case studies help to bring out the comparative features of the two models.

Readers are also asked to have a look at the Preface, which explains the general layout of the book.

Contents

PART I: FOUNDATION

1 Introduction

A modern computerised database is essentially a large volume of data stored with its description. Its concept was not a sudden breakthrough in computer technology, it was brought about rather gradually through experience and the pressure of requirements. Many organisations contributed in the development of databases; the list is long and includes computer manufacturers, software houses, user organisations and professional bodies, as described later in this chapter. The chapter also covers some basic concepts, and advantages and disadvantages of databases; but we begin with the history of the evolution of databases.

1.1 History of the evolution of databases

A database may be regarded as the most modern technique of data storage which started with the invention of punched cards by Dr Herman Hollerith (Hollingdale and Tootil, 1970) of the United States Bureau of Census in the 1880s. Dr Hollerith was confronted with the problem of completing the 1880 census of 13 million Americans before 1890 when the next census was due. In 1886 it was clear that the job could not be completed in time using the existing manual means of counting. Necessity was the mother of invention: from his knowledge of the use of punched cards in Jacquard looms, Dr Hollerith invented a method of storing information on them, thus introducing the era of mechanised card files which was to remain the leading information storage medium for the next sixty years.

The first electronic computer ENIAC became operational in 1946. It was designed by Professors Eckert and Mauchley of the University of Pennsylvania for the United States Defense Department, mainly to calculate trajectories and firing tables. In those early days, computers were largely used for scientific calculations where the facility for the storage of data did not feature as an important requirement – the speed of arithmetic calculation was all that was needed. But when their usage was subsequently extended to data processing, the limitations of card files began to be noticed. One of the organisations which became seriously concerned by such limitations was the U.S. Bureau of Census. Faced with the approaching 1950 census, the Bureau became particularly anxious to have a faster storage medium, and this led to the invention of magnetic tape devices. In 1951 a new computer, to be called Univac-1, designed by the same Echert-Mauchley pair, was delivered to the Bureau to cope with the 1950 census data. It had a unique device called

Magnetic Tape System which could read a magnetic tape both forward and backward at high speed (Rosen, 1969). The necessity of the Census Bureau thus mothered two major inventions in seventy years – the punch card and magnetic tape files.

The impact of magnetic tape on data storage was overwhelming. Those of us involved in some early card file processing can still recall the frustration of finding mispunched, slightly off-punched and screwed up cards scattered in large card files produced by a computer. The fear of accidentally dropping a box of cards and getting the cards out of sequence was also considerable. With the advent of the magnetic tape, all these nightmares were over. It was light, reliable and neat; its storage capacity and speed were phenomenal compared to a card file. However, the magnetic tape did not alter the processing mode substantially. The file organisation was still sequential, although it allowed the representation of variable length records in a more convenient form. All terminologies of card file systems such as files, records and fields were carried over to the magnetic tape system. The data-processing systems used in the 1950s were mostly simple payroll subsystems designed in isolation and independent of other related subsystems. A typical payroll subsystem would consist of a large number of small programs with many files, each containing fragmented information. The situation changed when the subsequent attempt to computerise more complex systems brought in a new breed of experts – the system analysts. They took a global view and introduced the concept of integrated files to be shared by a number of programs in more than one subsystem. These shared files were relatively large and the problem of writing long data descriptions in each program for each file was resolved by the Cobol COPY verb which allowed a program to copy a pre-written general data description of a file. Conceptually the introduction of the COPY verb represented a major step forward in the separation of data description from any program. From then on the history of the evolution of database became an account of the progress towards more integrated storage of data from less integrated storage.

The introduction of magnetic discs in the mid sixties gave a further boost towards this integration. To access a record on a magnetic tape it is necessary to scan all the intervening records sequentially, but on a disc a record can be accessed directly, by-passing the other records, and thereby gaining an overall retrieval speed of 2 to 4 orders of magnitude over magnetic tape. Disc storage, thus, provided the much needed hardware support for the large integrated files.

By the mid sixties, the concept of Management Information System (MIS) gained currency. The basic approach was to run the programs of the MIS package on the files output by all the relevant subsystems, but soon it was found that, for a large organisation, the number of input files to the MIS package was excessively high with the attendant problems of extensive sorting and collating. Additionally the failure of one system could easily wreck the whole operation. The data duplication in the files resulting in update inconsistencies brought in another problem. Thus these MIS packages turned out to be unreliable, cumbersome and generally unsatisfactory. This highlighted the need for greater integration and induced many organisations to favour such a measure.

One of the outstanding products of that time was the Integrated Data Store (IDS) released by General Electric (now owned by Honeywell and called IDS-I) in 1965. As the name suggests, IDS was used to create large integrated files to be shared by a number of applications. It is a forerunner of modern database management systems and is capable of supporting a number of data structures. Its pioneer, Charles W. Bachman subsequently played a very active role in the development of the Codasyl database proposal which incorporates many of the features of IDS.

IDS was soon followed by other MIS packages based on integrated files for major systems. Many organisations invested large sums of money on them only to discover that MIS packages were not as effective as they would like them to be. The problem was the lack of coordination between the files of the major systems. It was soon realised that what was needed was a database containing a generalised integrated collection of data, ideally for all the systems of an organisation serving all application programs. It was recognised that such a database must be both program and language independent if it was to serve all applications; and in particular a change in the data should not require a change in the application program. If a database is to respond efficiently to the conflicting needs of all the application programs then it must provide an adequate data representation facility – supported by a variety of data access techniques. This concept of a database crystallised only in the early seventies although the term database or data bank has been used loosely from the mid sixties to refer to almost any large file.

A number of database management systems, based on diverse data models, appeared on the market in the early seventies. Codasyl became interested in databases in the late sixties and set up a task group to specify a common data model, now known as the *Codasyl* or *network model*. Since the publication of its draft specifications in 1971, there has been a noticeable movement away from the diversity of earlier models, many implementors converging on the Codasyl model.

In parallel with the development of the Codasyl model, other ideas based on mathematical concepts were also pursued by computer scientists. The major outcome of this search was the relational model due to Dr Edgar F. Codd which was first proposed in 1970. The model is simple, elegant and yet very powerful. It caught the imagination of researchers early on, and has been the principal inspiration behind the new area of database research. If Bachman is the father of databases, then Codd is the father of database research. The earlier implementation problems of the relational model have now been resolved, and there now exists a number of relational implementations, including commercial products and research prototypes. Today, the relational model is recognised as the other major model beside the network model. There is however a growing realisation among the proponents of the two models that what is needed is a unified approach where the facilities of both the models, indeed all models, are available within a single framework. This has led to research in data modelling.