Data Analysis for Data Base Design

D R Howe

Data Analysis for Data Base Design

D. R. Howe

Principal Lecturer in Data Processing
Leicester Polytechnic

Edward Arnold

First published 1983 by
Edward Arnold (Publishers) Ltd,
41 Bedford Square, London WC1B 3DQ

Edward Arnold
3 East Read Street
Baltimore
Maryland 21202
U.S.A.

Edward Arnold (Australia) Pty Ltd
80 Waverley Road
Caulfield East
Victoria 3145
Australia

Reprinted 1984, 1985, 1986

British Library Cataloguing in Publication Data

Howe, D.R.
Data analysis for data base design.
1. Data Base Management
I. Title
001.64'42 QA76. 9.D3

ISBN 0-7131-3481-X

Printed in Great Britain by
Thomson Litho Ltd, East Kilbride, Scotland

Preface

Readership

Over the past decade or so, the subject of data analysis for data base design has blossomed into a tool of great practical value to the systems analyst and designer.

This book is intended to make the techniques of data analysis more readily available to the student of systems analysis. Most modern texts on systems analysis and design include some mention of data analysis techniques, but there is a dearth of material which examines the subject both in detail and from a practical rather than a mathematically abstract viewpoint. It is assumed that the reader has some background knowledge of data processing, but no more than would be expected from a first year Higher National Diploma or Degree course in computer studies.

Scope

In keeping the book within reasonable logical and physical bounds I have applied the following criteria.

Firstly, the book is about data analysis, not systems analysis or data processing management. The reader seeking an explanation of how data analysis fits into a wider perspective should consult a text such as Parkin's book on Systems Analysis.[1]

Secondly, although it is scarcely feasible to discuss data analysis without propounding some kind of methodology, I have kept in mind that my aim is to help the reader think about the problems and obtain a grasp of the basic tools, rather than to grind the axe of any particular methodology. For example, Chen's entity-relationship model is the basis for the discussion in Part 3, but no claim is made that this is an exposition of Chen's ideas; indeed I have freely selected and adapted his ideas to suit my own purpose. This book will have served its purpose if it leaves readers better equipped to assimilate the ideas in any of the present or future 'brand-name' methodologies, or to devise methodologies of their own.

Thirdly, I have deliberately restricted the scale of the applications under consideration in the examples and exercises. It is important that the reader should tackle more substantial problems, but I believe that this is best done under some form of tutorial guidance, either on a course or on-the-job, as the ramifications of large-scale, or even medium-scale, applications are inevitably extensive. In this respect the role of the

[1]References and comments in this preface may be followed up via the annotated bibliography at the end of the book.

book is to free the tutor for more of this type of work. As a corollary, I have excluded topics for which it would be difficult to make a convincing case in the context of small-scale applications. For instance, the use of a computerised data dictionary may be indispensable for a sizable system, but the need for it is singularly unconvincing when one is faced with a mere handful of data items. As for documentation standards in general, practice and preference vary so much that I have not ventured further along that path than seemed necessary to my immediate purpose.

Structure and content

The book is divided into five parts:

A: Introduction
1: Data bases and data base management systems
2: Relational modelling
3: Entity-relationship modelling
4: Implementation

Part A differs from the others in its 'case history' style. The reader who is only too familiar with the problems introduced here may wish to skip through this part fairly quickly.

Part 1 examines the concepts of *data base* and *data base management system* (DBMS). A DBMS is viewed not so much as a good thing in its own right, but more as a means of overcoming the problems of shared data. The architecture of a DBMS is discussed in terms of the ANSI/SPARC model; it is this architecture around which the discussion on implementation in Part 4 is organised.

Part 2 explains how a data model can be designed in a 'bottom-up' direction via the application of normalisation techniques to the data attributes. The usual way of treating normalisation is to beat a trail down the first, second, third, fourth, and maybe fifth, normal form route, with Boyce/Codd normal form snapping hard at the heels of third normal form. I cannot see the merit of this approach, which seems to have become prevalent for historical reasons rather than for its intrinsic worth, and I have a sneaking suspicion that it is sustained chiefly by its convenience as a source of examination questions. My approach has been to distinguish carefully between *duplicated* data and *redundant* data, and hence establish the Boyce/Codd rule directly. Tables (i.e. relations) which satisfy the Boyce/Codd rule are described as *well-normalised*. Even well-normalised tables may still contain redundant data; once this is eliminated the tables are said to be *fully-normalised*. Because the Boyce/Codd rule is stated in terms of the concept of a determinant, I often use the terms *determines* and *determinancy* where it might be more usual to find the opposite viewpoint being taken, coupled with the use of terms such as *is dependent on* and *dependency*. I have not dealt with fifth normal form explicitly as, although theoretically interesting, it is of little practical significance.

Part 3 builds on the results of Part 2, but takes an opposite 'top-down' approach which starts by identifying entity and relationship types, and then uses these to construct a framework into which the attributes may be slotted. Part 3 concludes with a discussion of how the design may be 'flexed' to improve its performance.

Part 4 shows how an entity-relationship model can be implemented in terms of the Burroughs DMS II and Codasyl data base management systems. Burroughs DMS II is

chosen as one vehicle, partly because it offers an interesting contrast to the Codasyl approach, and partly because the implementation technique described is easily adapted to any system which allows multiple indexes for files, in particular the 1974 American National Standard for COBOL. Codasyl is chosen as the other vehicle because its influence on data base thinking is so pervasive, as evidenced by the large number of implementations of Codasyl-style systems. The Codasyl treatment is based primarily on the 1981 Journals of Development prepared by the Codasyl Data Description Language Committee and the Codasyl COBOL Committee, but some references to the 1978 Codasyl Data Description Language Committee Journal of Development are made, both because it is instructive to see how the system has continued to move closer to the ANSI/SPARC architecture, and also because implementations may lag several years behind the current specification. Although there was a temptation to include reviews of other DBMS, it would not have been possible within the scope set for the book to do more than scratch at the surface of each, which might have created the illusion of thorough coverage but would have done little to inform the reader. The only exception I have made in this connection is in the final chapter, where the idea of a relational data manipulation language is introduced as a contrast to the COBOL-based procedural languages discussed earlier. A full treatment of relational languages and data base management systems would deserve a book of its own. Part 4 draws heavily upon Codasyl Journals of Development and Burroughs Corporation reference manuals. Readers who intend to use a Codasyl-based system or Burroughs DMS II should consult the current editions of the relevant manuals to verify the applicability to their system of the information given here.

Questions and assignments

Merely reading about a subject does not of itself confer much practical skill or insight. Consequently, the text is interspersed with a large number of questions and several assignments; these frequently offer more than just routine practice, for they may amplify, anticipate or challenge the text, and together with the answer pointers at the end of each chapter they form an integral part of the book. The reader is assumed to have at least read the questions and answer pointers. For those with the resolve to tackle the questions (those who do not will surely reap their just reward!) estimated times are given for each question. These are 'thinking times' rather than 'writing out model answer times', and may be used in various ways. For example:

(i) Ignore them. This is the best course of action if you are irritated at the thought of some kind of ectoplasmic author hovering over you with a stopwatch.
(ii) If, on a first reading, you have not made any progress towards an answer within the allotted time, look up the answer pointer.
(iii) On a second reading (assuming you have the stamina) try to sketch out the main features of the answer within the time stated.

The answer pointers should not be regarded as infallible model answers, but as aids to thinking about the questions. Some answer pointers do not cover all the points raised in a question, whereas others may offer additional commentary or even raise further questions. The assignments offer more substantial tasks which may be found useful for seminar, or project work.

Terminology

The terminology and diagrammatic conventions used deserve some comment. The term *table* has been preferred to *relation* for several reasons. Table is more meaningful at a first encounter with the subject, it avoids confusion between relation and relationship, and it evades the barbarity of *relationship relation* in the context of the entity-relationship model. Having studied this material, the reader should have little difficulty in adapting to the use of relation instead of table, so I trust that those who know the difference between a table and a relation will forgive my lapse from grace. In the discussion of entity-relationship modelling I have used my own terms *obligatory* and *non-obligatory*, where others use *mandatory, optional* and *contingent.* This is done to eliminate confusion with the related, but distinct, meanings attached to mandatory and optional in the context of the Codasyl data base proposals, and because *contingent relationship* is an ambiguous term which fails to indicate which entity type has obligatory membership, and which non-obligatory. The inelegance of non-obligatory is, I think, outweighed by its clarity. Chen's notation (1:1, 1:N, M:N) for representing the degree of a relationship is preferred to the 'crowsfoot' notation, used for example in the National Computing Centre's documentation standards, mainly because the latter does not permit the drawing of an 'uncommitted' diagram in which the relationships are shown but not their degrees. As entity-relationship diagrams are as much a tool for thinking about the problem as for presenting a final design, this inability to represent an uncommitted diagram is distinctly inconvenient. Similarly, the convention that solid and dashed lines represent, respectively, obligatory and non-obligatory membership of an entity type in a relationship, does not permit the drawing of a diagram which is uncommitted with respect to membership class, so I have used my own 'blob' notation which, as it happens, is easier to draw anyway. I have done my best not to abuse the English language, but regret that I bring no solace to those who, justifiably, maintain that data is plural and schemas is non-existent.

Acknowledgements

It is a measure of the contribution of the pioneers in the data base field that a book of this nature takes much of their work for granted. Their ideas have become conventional wisdom to the extent that I feel justified within a tutorial text in not referencing primary sources explicitly. An annotated bibliography is included at the end of the book as a guide to further study. By following up these references it will soon become clear who deserves credit for what.

Finally, my personal acknowledgements. John Darby, now at Teeside Polytechnic, was a valued mentor in my formative days in computing at Rolls-Royce Ltd. Many colleagues and students at Leicester Polytechnic, and (through my work as a Course Tutor) at the Open University, have contributed directly or indirectly to the preparation of this book. In particular, I would like to thank Steve Skidmore for reviewing the earlier chapters, and Gillian Mills, Andrew Parkin and Christine Warner for their constructive comments on the final draft. Marilyn Hill, our Academic Librarian, showed much ingenuity in pursuing a particularly elusive reference. The final result is, of course, my responsibility. Special thanks are due to Andrew Parkin for

encouraging me to start (and finish) the book, for going to Australia for eighteen months so that I had to think for myself, and for achieving through his active promotion of the subject a widespread acceptance of data analysis as being an important component of our courses. He has done much to create a thriving Systems Analysis Teaching Group at the Polytechnic.

My wife, Christine, and sons Richard and Geoffrey have been more understanding of the demands of authorship than I had any right to expect, and I thank them for their patience. Christine also made an excellent job of typing the manuscript in between caring for an assortment of ponies, ducks, chickens, goldfish, a dog, a cat and a guinea pig, not to mention two boys and a husband.

David Howe
Ashby-de-la-Zouch
1982

Acknowledgements to CODASYL

A product of CODASYL is not the property of any individual, company, or organisation or of any group of individuals, companies, or organisations.

No warranty, expressed or implied, is made by any contributor or by any CODASYL committee as to the accuracy and functioning of any system developed as a result of this product.

Moreover, no responsibility is assumed by any contributor or by any committee in connection therewith.

This product neither states the opinion of, nor implies concurrence of, the sponsoring institutions.

COBOL is an industry language and is not the property of any company or group of companies, or of any organisation or group of organisations.

No warranty, expressed or implied, is made by any contributor or by the CODASYL COBOL Committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

Acknowledgement to Burroughs Corporation

DMS II is a trademark of the Burroughs Corporation.

Contents