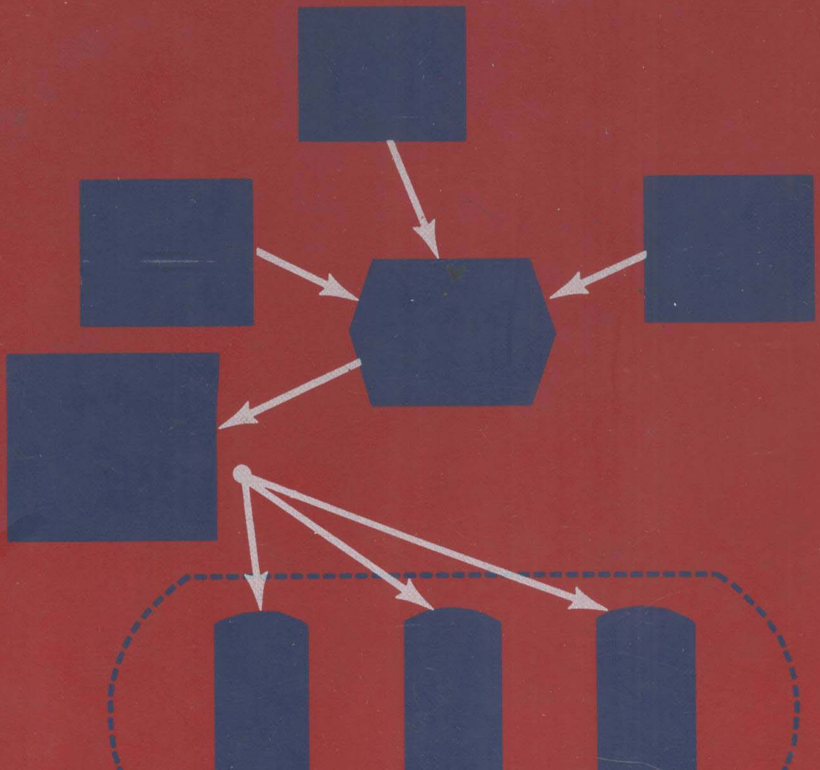
EFFECTIVE DATA BASE DESIGN

William H. Inmon



PRENTICE-HALL SERIES IN DATA PROCESSING MANAGEMENT

Leonard Krauss, *Editor*

EFFECTIVE DATA BASE DESIGN

WILLIAM H. INMON

PRENTICE-HALL INC.
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging in Publication Data

INMON, WILLIAM H
Effective data base design.

Bibliography: p.
Includes index.
1. Data base management. I. Title.
QA76.9.D3I54 001.64 80-15025
ISBN 0-13-241489-9

PRENTICE HALL SERIES IN DATA PROCESSING MANAGEMENT
Leonard Krauss, editor

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

Editorial Production/Supervision by Theodore Pastrick
Manufacturing Buyer: Anthony Caruso

For my dad, Garland Inmon

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL OF CANADA, LTD., *Toronto*
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

The motivation for writing *Effective Data Base Design* stems from the curious phenomenon of consulting with many data base users and observing over a period of time a recurring pattern of the very serious problems encountered in the building of data base applications. The discussions with various data base users ranged from in-depth to superficial. In the vast majority of cases, the problems of their systems could be directly traced to design: system design, data design, and in some cases program design. In the case where program design alone was the problem (a rare phenomenon) other books on programming, modularity, structured walk-throughs, and so on, apply. In the more usual case, a poor design of data structures, data bases, and interrelationships of data, was at least a contributing factor in the building of unsatisfactory systems and was usually a large part of the problem. It became apparent that solving the major problems of data design would be a significant step toward the larger problem of building satisfactory data base applications.

It is a sad fact of life that when a person learns the mechanics of data base design, he or she often feels equipped to design an application with no further study. Perhaps the architects of today's data base management systems have done a disservice by making it too *easy* to construct data bases. The freedom the beginner has in constructing data bases usually results in a poor design, which in turn triggers all sorts of other problems. The purpose of this book is to explore many of those problems and to discuss how they can be prevented.

In many respects, the most critical moment in the life of the development of a system is its design. Once the designer casts the form of the data "in concrete," it becomes increasingly expensive (in terms of labor, money, machine resources, etc.) to change the design. The longer the system development process goes, the more expensive and distasteful change becomes, however the system is designed. This hard fact of life is usually learned by management after one (sometimes two) really expensive failures. The philosophy of "build it now, make it run later" has proven over and over again to be an open invitation to all types of failure.

This book attempts to do two things: to identify the major problems of data base design in today's production systems, and to discuss solutions to those problems that are appropriate at the design level and, in some cases, at the implementation level. The problems identified (in one form or the other) are typical of nearly all production data base applications. The author does not assume an understanding of IMS by the reader in the identification of data base problems. In discussing the solution to data base design problems at the design level, some discussions are relevant to IMS, others are not. In discussions of problems at their lowest level, the implementation level, some knowledge of IMS is quite useful.

The important points to be made that are couched in terms of IMS have equal validity in one form or the other in most other data base management systems. The interested reader should have little difficulty in translating those points when appropriate. The intent of this book is not to teach the reader IMS. The intent is to alert the reader to data base design concepts and problems and to discuss solutions.

The book attempts to address both the academic and the industrial reader. Different types of readers from industry will find parts that will interest them. Management will be interested in concepts of data base design. Data administration personnel will be interested in such topics as design review methodology and auditing of data bases. Data analysts will be interested in the performance of the application. Application personnel will be interested in the review checklist for data structures.

In addition to data base design discussed at the conceptual level, the academic community will be interested in such topics as data elasticity and the achievement of flexibility in data base design. There is a rather direct relationship between the analytical tools for flexibility and canonical data structures, for example.

Chapter 1, which deals with the evolution of data base design, is of particular interest. Because many of us deal with data bases and data base design on a day-to-day basis, we are too close to our work to perceive the nearly universal evolution that is occurring. Often, stepping back and viewing data base design and development from a different perspective leads to powerful new insights.

Chapter 5 addresses flexibility in the structuring of data. A careful reading of this chapter will arm the reader with analytical tools that will allow him or her to assess the degree of flexibility of a data structure early in the design phase of the system, at the point where change can be made to the data structure without severely impacting the system.

Chapter 7 addresses problems associated with large data bases. It is likely that new data base management systems (and new releases of existing systems) will have features that will ease some of the problems that exist today. With today's software, the designer is on shaky ground when the size of the data bases he or she is working with becomes unwieldy.

Chapter 8 discusses the restructuring of data bases. By means of introducing the user to some untraditional techniques of design, the author attempts to free the reader from some of the concepts of "classical" data base design. Often, elegance at this level pays dividends in many other places.

Chapter 10 discusses the structural analysis of data by means of a checklist. The list is a start toward tying down design review criteria diagnostics in a formal manner.

The exercises at the ends of some of the chapters are intended to be their own reward. For the most part there are no "right" or "wrong" answers. Instead, the benefit of the exercise will come to the reader in pursuing the activity suggested. It has been the author's experience that nothing replaces active participation.

The main thrust of this book is toward problems associated with the design and construction of production data base systems. Experimental or research-oriented data base management systems are not emphasized. In the future it is likely that relational data base management systems will make their presence felt in the world of production data base systems. When they do, they will inevitably carry with them their own set of problems (which may not be very different from those presented in this book). When that time arrives, it will be appropriate to address them in the same way that we address the data models that comprise the majority of production systems today.

ACKNOWLEDGMENTS

This book has been influenced technically by two people, Mark Sherman and Jim Sheetz. Mark's influence is found throughout the book. His contributions come mainly from conversations and work experiences. Jim's influence has come mostly through his presentations on such subjects as design review methodology and the development of on-line systems.

Also, Jeanne Friedman has been invaluable in her editing and commentary, both of which have helped to produce a better final product.

For these reasons I am grateful to Mark, Jim, and Jeanne.

W. H. Inmon

CONTENTS

ONE

EVOLUTION OF DATA DESIGN

As a data processing installation matures, its needs for and priorities of design evolve. There is a rather predictable course of evolution that typifies most (not all) shops as they grow over time. Generally speaking there are four recognizable stages of evolution: design by default, design for the introduction of the data base concept, design for performance, and design for flexibility (Fig. 1.1).

DESIGN BY DEFAULT

The first level of design, *design by default*, is typified by processing flat files. The media associated with this level of design are sequential tapes, paper tapes, sorted cards, and some direct-access usage, such as ISAM (indexed sequential access method). Data is processed in batch mode almost exclusively. The designers at this level are faced with problems of large sorts and merges, multifile match programs, card editing and process-

$$\begin{tikzcd}\text{Design by default} \arrow[r] & \text{Data base concept} \arrow[r] & \text{Data performance} \arrow[r] & \text{Data flexibility}\end{tikzcd}$$

Figure 1.1 Evolution of Data Design: Data design has evolved through four stages within a mature organization.

ing, and voluminous reports. Figure 1.2 depicts the environment of the designer.

Some of the major problems with design by default are redundancy of storage, redundancy of processing, an exponential increase in complexity as systems grow in volume and as they age, and inability to change the system as its real-world representation changes. The availability of data to the user is an issue because as new systems are created or changes to old systems are made, the cost of change and new development is high and programs often take a long time to implement. Data processing departments at this level of design often have the reputation of being sluggish and unresponsive to the user. Users learn not to expect expedient results. Often, by the time a new system or major changes to a system are

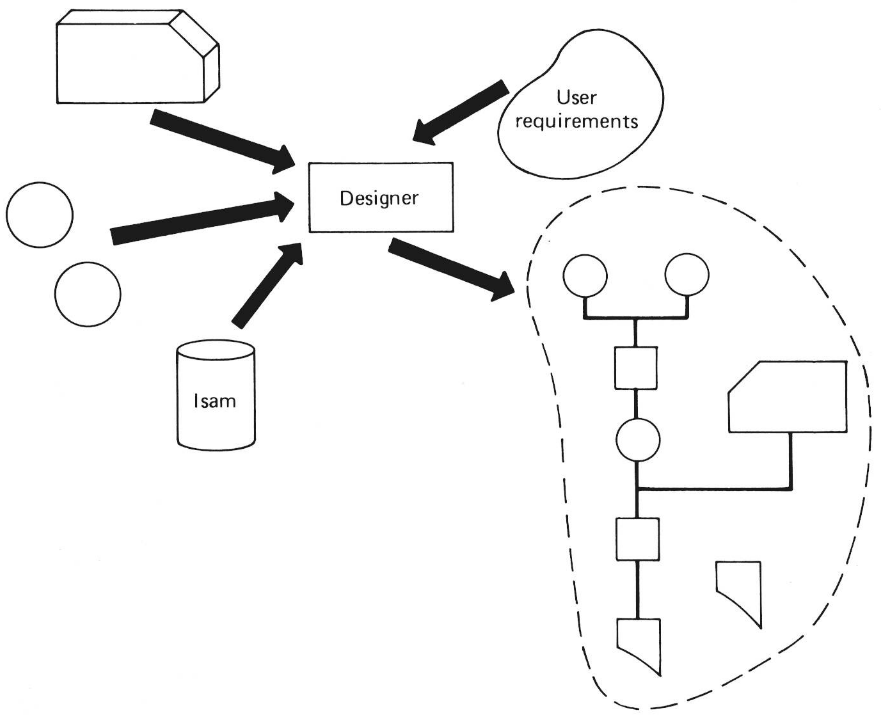


Figure 1.2 Evolution of Data Design: Design by Default. System development is usually a long and tedious process. Design is dominated by flat file concept.

implemented, the users' needs have changed so that the system as initially designed no longer fits the users' requirements. The exponential demand for new systems and the corresponding growth in system size leads to the next step of evolution—the data base concept.

DATA BASE CONCEPT

The *data base concept* evolved as an answer to the problems of design by default. Shops implementing the data base concept did so to reduce wasteful processing, reduce wasteful storage, centralize data and increase user availability, enhance control of data (physically and conceptually), and allow the development of systems to continue in an orderly fashion. Ultimately the concept of data base achieved some, but certainly not all of the goals the original enthusiasts had envisioned. Going "data base" did not prove to be the panacea it was originally advertised to be.

To achieve the data base concept, it is necessary to have a *data base management system* (DBMS). Most shops purchased the software; a few shops built their own. It is through the DBMS that the user can access data and derive the benefits of the data base concept. A major selling point of the data base concept is the ability to share data (or integrate it) and the freedom it allows the designer.

The designer can usually choose from many options of the DBMS to implement the same functional result. DBMSs are necessarily complex because they have to take into account many diverse and complex features, such as the teleprocessing network that used the DBMS, the physical storage of data, the logical views of data superimposed on the physical data, the needs of application programs to process the users' requests, the vagaries of the SCP (system control program) under which the DBMS runs, and operational aspects of the running of the system. In short, the greater the function of the DBMS, the more complex it becomes. Furthermore, the environment it manages is constantly changing. This makes the job of the DBMS even more difficult. Figure 1.3 depicts some of the complexities the DBMS must deal with.

The initial thrust of data base designers was to learn what tools were available within the DBMS and how to use them. In doing so, some of the potential of the data base concept was unlocked. There was a problem, however, in building data base systems knowing only what tools the DBMS offered and the mechanics of using them. The vast majority of the systems designed and built in this mode are terribly inefficient. In the case of batch systems, inefficiency may be tolerable or manageable. In the case of on-line systems, the inefficiencies of poor design are usually unacceptable for no other reason than high visibility. In fairness, as data

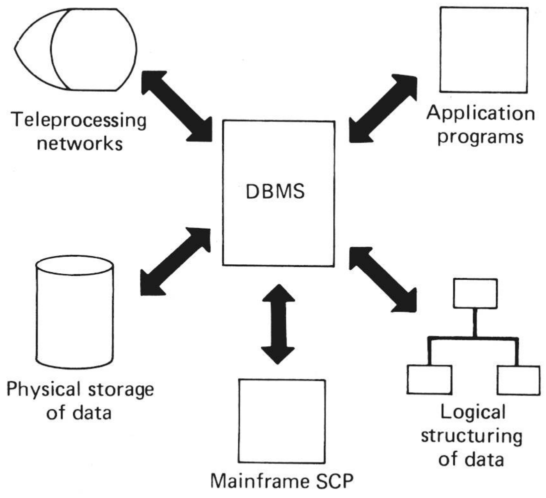


Figure 1.3 Evolution of Data Design: Data Base Concept. Data base management systems are complex because they manage many loosely related aspects of a computer system and a company's loosely related information requirements. Typically, the DBMS gives the designer many ways of achieving the same functional result.

base designers are taught the tools of using the DBMS, they are not usually taught the economies underlying use of the tools. It is a normal occurrence to have two very similar options perform in drastically different ways.

As shown in Fig. 1.4, the designer in the data base environment must

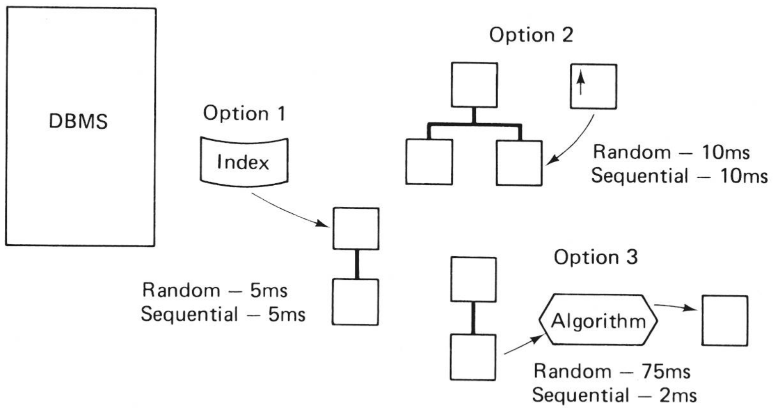


Figure 1.4 Evolution of Data Design: Design for Performance. To design for performance, the designer had to understand the economies involved in the selection of options available within the DBMS. Understanding the options allowed the designer to make quantifiably justified trade-offs. More sophistication on the part of the designer was required.

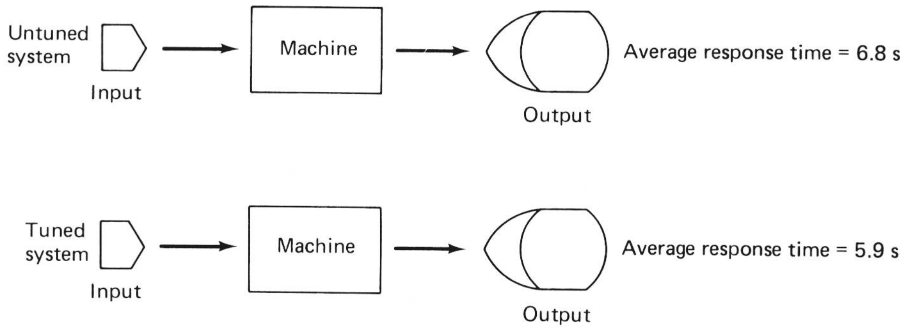


Figure 1.5 Evolution of Data Design: Design for Performance. Shops that did not design for performance found that tuning a system helped, but was not the secret to performance.

understand the significance of his or her choice of design options. Unless the designers have been forewarned about certain practices, the result is an innocently designed failure. With all the options that are available, the designer has to cope with some very complex and tedious problems based upon scanty, cryptic, or nonexistent information. It is then no wonder that systems designed at this level typically perform poorly.

Managers found that buying bigger and faster hardware was not a solution to performance as it improved performance only marginally. This was a surprise because in the era of design by default, buying upgraded hardware was always an alternative to greater throughput. Managers also found that tuning their systems produced only marginal results. In short, if the design had not been done with performance objectives understood from the very start, there was not a lot the manager could do except to rebuild the system.

Figure 1.5 illustrates the fact that marginal performance can be obtained by tuning the system. Tuning is a constant effort, since the size of data bases, the transaction volume and mix, the degree of organization of data, and many other relevant factors are constantly growing.

Figure 1.6 shows that adding machine speed and capacity is not the solution to achieving satisfactory on-line systems.

DESIGN FOR PERFORMANCE

The first step in *designing for performance* is to understand the underlying economies involved in choosing DBMS options. Because of the complexities of the DBMS, this is not always an easy thing to do. Once the economies are understood, there are several steps that can be taken. The