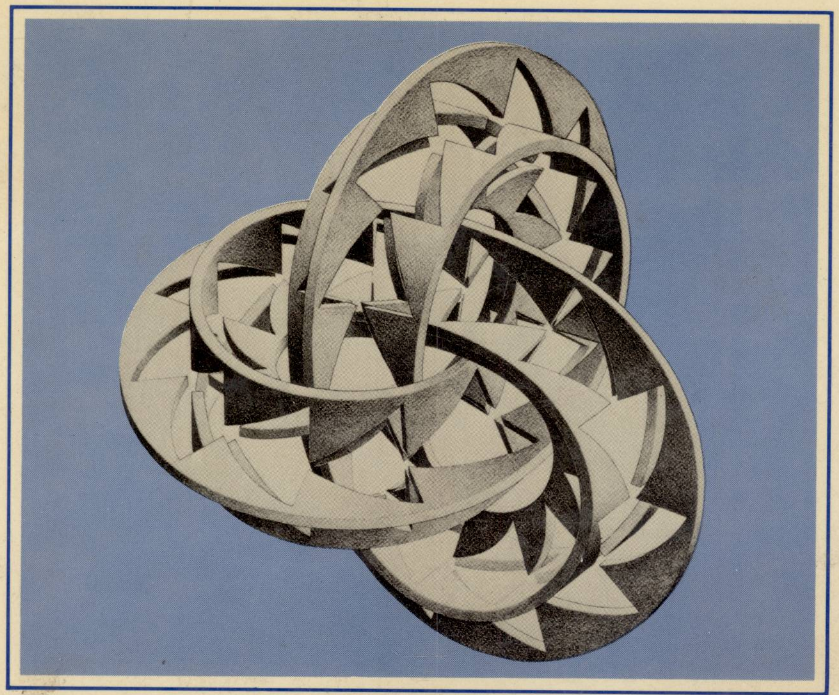
DATABASE DESIGN FUNDAMENTALS

NAPHTALI RISHE



DATABASE DESIGN FUNDAMENTALS

A STRUCTURED INTRODUCTION TO DATABASES AND A STRUCTURED APPLICATION DESIGN METHODOLOGY

NAPHTALI RISHE

Florida International University

PRENTICE HALL, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Rishe, Naphtali.
Database design fundamentals.

Bibliography.
Includes index.
1. Data base management. 2. Electronic data processing
—Structured techniques. I. Title.
QA76.9.D3R55 1988 005.7 87-11443
ISBN 0-13-196791-6

Editorial/production supervision: Carolyn D. Fellows
Cover design: George Cornell
Cover art: M. C. Escher, *Knots*, 1966
© M. C. Escher Heirs
c/o Cordon Art
Baarn–Holland
Manufacturing buyer: Gordon Osbourne

To Bella and Rudolf Rishe

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-196791-6 025

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
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*
Simon & Schuster of Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

PREFACE

Uses of this book

- an undergraduate course on databases;
- a college course for future systems analysts and programmers;
- an "updating" course for software engineers, systems analysts, and programmers;
- suitable for a self-educating professional;
- a supplement for a graduate course on databases;
- a quick-reference manual and glossary of database terminology.

Knowledge to be gained

- the logical aspects ("the externals") of databases and database management systems;
- how to design and develop a high-quality database for any given enterprise;
- how to to select an appropriate database management system;

- how to program in the database environment;
- how to solve information access problems by use of database languages without programming;
- how to comprehend easily and critically the user manuals of commercial database management systems.

Most other textbooks devote their primary attention to the "internals" of database management systems and to theoretical aspects of databases. Neither of these issues is among the practical goals of the majority of the students, because they are future *users* of database management systems. They are users at the logical level of databases: *application* software engineers, systems analysts, and logical database designers, rather than developers of database management *systems* or theoreticians. Thus, the most important issues of the database study are:

- software design methodology for databases and programs, and
- application-world-oriented comprehension of database concepts.

Database design methodology

A novel methodology for logical design of databases is presented in this textbook. In the first step, a conceptual description of an enterprise is designed using a semantic binary model. Then, this description is converted into the relational, network, or hierarchical database design, in a form suitable for commercial database management systems. A high-quality database is produced as a result.

Chapter 4 explains why this methodology is significantly easier than and superior to (produces databases of higher quality) the methodologies based on the relational database theory, such as the normalization methodology. The normalization methodology used to be popular in the academic world, but was not practical enough or advantageous enough to be widely accepted by the industry.

Database models

The Relational, Network, and Hierarchical database models and languages are presented in this text as restricted forms of the semantic model in which the initial conceptual description of the user's enterprise is done. This reduces the reader's effort by minimizing the number of concepts to be learned. Although the different models have different terminology, the concepts are similar. After introducing the concepts, the text translates them into the terminology of each model. Most other textbooks introduce the terminology of each database model without relation to the other models. That causes quadruplication of the number of concepts to be understood by the student.

Database languages

Two classes of database languages are studied in detail. They are represented by two abstracted model-independent languages:

- a fourth-generation structured extension of a structured third-generation programming language (*Pascal* taken as an example), and
- a non-procedural language based on Logic.

These abstracted languages are used in all the database models to comprehend the specific languages of those models and of their database management systems. Several languages which are not strictly within the above classes of languages, but yet are very widely used in some database models, are also presented in this text. These languages are: SQL, Relational Algebra, and the CODASYL network navigation language. The presentation of SQL is very extensive. This text can be used as a reference manual and a user guide for those languages.

Prerequisites

Structured programming is required, preferably in Pascal or a similar language. Those who do not know Pascal or a similar language may wish to skip the sections on data manipulation extensions of programming languages. Those sections are not prerequisite to the other sections, and the other sections do not use Pascal.

No knowledge is needed of file organization or data structures.

Structure of the book

The book is composed primarily of explanations of concepts and examples. The examples are offset and boxed, so that the experienced reader or browser can easily skip them.

The concepts being defined are set in bold face. They are also referenced by the index.

The examples constitute a continuous case study of one application, for which databases are designed in different models, application programs are written in different languages, *etc.*

Most sections are followed by problems. Many of the problems are solved in the last chapter of the book. Page-number pointers direct the reader from the problems to their solutions.

If after reading a chapter the reader fails to solve a problem marked 'Advanced' or 'Optional', it does not mean a lack of understanding of the chapter. (It rather means the lack of mathematical knowledge or experience, which is not prerequisite to the reading of this book.)

The sections marked with '*' contain optional, usually advanced, material, and may be skipped. Optional advanced material within the regular sections is given in the footnotes.

Acknowledgment

I am grateful to the Computer Science Department of the University of California, Santa Barbara, for providing facilities and an environment conducive to writing this book, and to the following students, whose comments on the manuscript were very helpful: Shih-Chao Chang, Changlin Chen, Wentsung Chen, Janine Felzer, David Galvin, Lionel Geretz, Alok Jain, John King, Nina Lewis, Ben Lipkowitz, Ernest Liu, Hemant Madan, Brigit Prochazka, Eric Rapin, Jack Schwartz, Donald Traub, and Narayanan Vijaykumar.

I thank the staff of Prentice Hall, especially Valerie Ashton, Carolyn Fellows, and Cynthia Scheel, for their excellent professional work.

I would like to express my deep gratitude to Professors Allen Reiter and Alan Konheim, whose guidance and support were of great value in my professional development.

CONTENTS

1 DATABASES AND SEMANTIC MODELING

"Database — From DATA and BASE (adj - low, mean, vile, etc). A place where data can be lost in a structured manner.

"DBMS (Database Management System) — The software needed to set up highly complex inter-relational data structures, so that files can be lost in any convenient sequence (e.g. Index before data; First-in-last-out)."

From a folklore dictionary.

This chapter introduces the basic concepts of databases and logical representation of real-world information in databases.

1.1. Databases

General-purpose software system — a software system that can serve a variety of needs of numerous dissimilar enterprises.

Example 1-1.

A compiler for Pascal.

Application — a software system serving the special needs of an enterprise or a group of similar enterprises.

Example 1-2.

The registration of students in a university.

Application's real world — all the information owned by, and subject to computerization in an enterprise, *or* all such information which is relevant to a self-contained application within the enterprise.

Example 1-3.

The examples of this text constitute a case study. Its application world is the educational activities of a university. The information contains

- a list of the university's departments (including all the full and short names of each department)
- personal data of all the students and their major and minor departments
- personal data of all the instructors and their work information (including all the departments in which the instructor works and all the courses which the instructor teaches)
- the list of courses given in the university catalog
- the history of courses offered by instructors
- the history of student enrollment in courses and the final grades received

Database — an updatable storage of information of an application's world *and* managing software, that conceals from the user the physical aspects of information storage and information representation. The information stored in a database is accessible at a *logical* level without involving the physical concepts of implementation.

Example 1-4.

Neither a user nor his program will try to seek the names of computer science instructors in track 13 of cylinder 5 of a disk or in "logical" record 225 of file XU17.NAMES.VERSION.12.84. Instead, the user will communicate with the database using some *logical* structure of the application's information.

Normally, a database should cover *all* the information of one application; there should not be two databases for one application.

Database Management System, DBMS — a general-purpose software system which can manage databases for a very large class of the possible application worlds.

Example 1-5.

A DBMS is able to manage our university database and also completely different databases: an Internal Revenue Service database, an FBI *WANTED* database, a UN database on world geographical data, an Amtrak schedule, *etc.*

Instantaneous Database — all the information represented in a database at a given instant of time. This includes the historic information which is still kept at that time.

The actual information stored in the database changes from day to day. Most changes are additions of information to the database.

Example 1-6.

A new student, a new instructor, new events of course offerings.

Fewer changes are deletions of information.

Example 1-7.

Historic information past the archival period;
a course offering which was canceled before it was given.

Some changes are replacements: updates; correction of wrongly recorded information.

> *Example 1-8.*
>
> Update of the address of a student;
>
> correction of the student's birth year (previously wrongly recorded).

Hence the life of a database can be seen as a sequence of instantaneous databases. The first one in the sequence is often the empty instantaneous database — it is the state before any information has been entered.

Database Model — a convention of specifying the concepts of the real worlds in a form understandable by a DBMS. (Technically, it is an abstract data structure such that every possible instantaneous database of nearly every application's world can be logically represented by an instance of that data structure.)

The following database models will be studied in this text:

- **Binary** (also called *Semantic Binary* or *Conceptual Binary*), in which the information is represented by logical associations (relations) between pairs of objects and by classification of objects into categories;
- **Relational**, in which the information is represented by a collection of printable tables;
- **Network**, in which the information is represented by a directed graph of records;
- **Hierarchical**, in which the information is represented as a tree of records.

The Binary Model is the most natural of the above models. It is the most convenient for specifying the logical structure of information and for defining the concepts of an application's world. In this text, the other models will be derived from the Binary Model. The Relational, Network, and Hierarchical models are dominant in today's commercial market of database management systems.

1.2. Categories

Object — any item in the real world. It can be either a concrete object or an abstract object as follows.

Example 1-9.

Consider the application world of a university.

I am an object, if I am of interest to the university. My name is an object. The Information Systems Department and its name "Information Systems Department" are two distinct objects.

Value, or **Concrete Object** — a printable object, such as a number, a character string, or a date. A value can be roughly considered as representing itself in the computer, or in any formal system.

Example 1-10.

My name and the name "Computer Science Department" are concrete objects. The grade 70 which has been given to a student in a course is also a concrete object.

Abstract Object — a non-value object in the real world. An abstract object can be, for example, a tangible item (such as a person, a table, a country), or an event (such as an offering of a course by an instructor), or an idea (such as a course). Abstract objects cannot be represented directly in the computer.

This term is also used for a user-transparent representation of such an object in the Semantic Binary Model.

Example 1-11.

The Management Science Department, the student of the department whose name is Alex Johnson, and the course named "Chemistry" are three abstract objects.

Category — any concept of the application's real world which is a unary property of objects. At every moment in time such a concept is descriptive of a set of objects which possess the property at that time.

Unlike the mathematical notion of a set, the category itself does not depend on its objects: the objects come and go while the meaning of the category is preserved in time. Conversely, a set *does* depend on its members: the meaning of a set changes with the ebb and flow of its members.

Categories are usually named by *singular* nouns.

Example 1-12.

STUDENT is a category of abstract objects. The set of all the students relevant to the application today is different from such a set tomorrow, since new students will arrive or will become relevant. However, the concept *STUDENT* will remain unaltered.

An object may belong to several categories at the same time.

Example 1-13.

One object may be known as a person, and at the same time as an instructor and as a student.

Example 1-14.

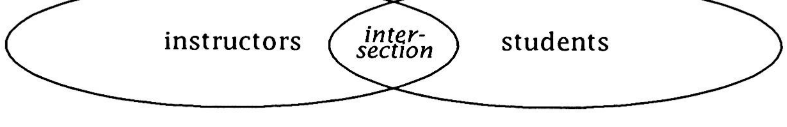
Some of the categories in the world of our university are: *INSTRUCTOR, PERSON, COURSE, STUDENT, DEPARTMENT.*

Disjoint categories — Two categories are *disjoint* if no object may simultaneously be a member of both categories. This means that at every point in time the sets of objects corresponding to two disjoint categories have empty intersection.

Example 1-15.

The categories *STUDENT* and *COURSE* are disjoint; so are *COURSE* and *DEPARTMENT* (even though there may be two *different* objects, a course and a department, both named "Physics").

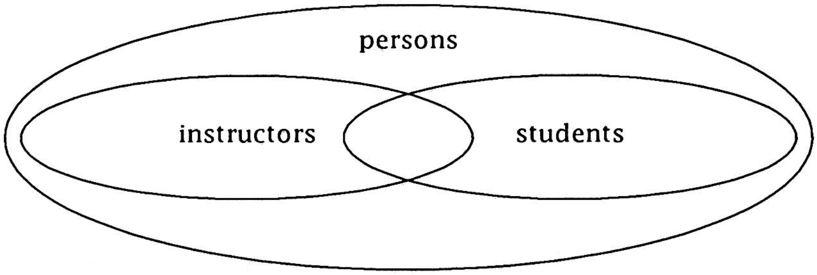
The categories *INSTRUCTOR* and *STUDENT* are not disjoint; neither are *INSTRUCTOR* and *PERSON.*



Subcategory — A category is a *subcategory* of another category if at every point in time every object of the former category should also belong to the latter. This means that at every point in time the set of objects corresponding to a category contains the set of objects corresponding to any subcategory of the category.

Example 1-16.

The category *STUDENT* is a subcategory of the category *PERSON*. The category *INSTRUCTOR* is another subcategory of the category *PERSON*.



Abstract category — a category whose objects are always abstract.

Concrete category, category of values — a category whose objects are always concrete.

Example 1-17.

STUDENT and *COURSE* are abstract categories. *STRING, NUMBER,* and *DIGIT* are concrete categories.

Many concrete categories, such as *NUMBER, STRING,* and *BOOLEAN,* have constant-in-time sets of objects. Thus, those concrete categories are actually indistinguishable from the corresponding sets of all numbers, all strings, and the Boolean values ({TRUE, FALSE}).

Finite category — A category is *finite* if at no point in time an infinite set of objects may correspond to it in the application's world.

Example 1-18.

The categories *STUDENT, COURSE,* and *DIGIT* are finite. The category *NUMBER* may be infinite.