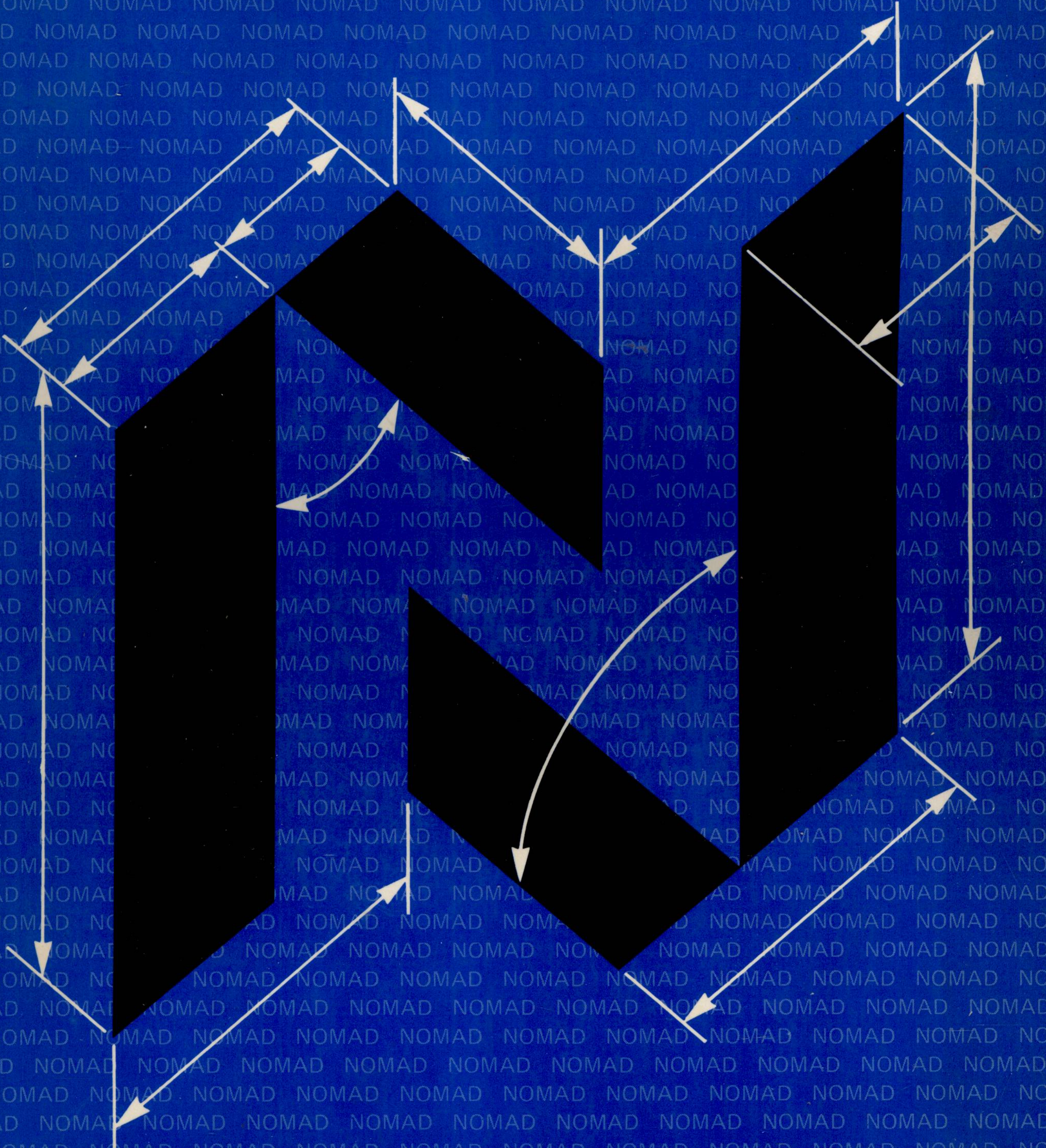
# A GUIDE TO NOMAD FOR APPLICATIONS DEVELOPMENT



by DANIEL D. McCRACKEN

# A Guide to NOMAD®
# for Applications Development

by

DANIEL D. McCRACKEN

ADDISON-WESLEY PUBLISHING COMPANY
Reading, Massachusetts
Menlo Park, California · London · Amsterdam · Don Mills, Ontario · Sydney

Library of Congress Catalogue Card No. 80-66545.

ISBN 0-201-04624-5
ABCDEFGHIJ-AL-89876543210

# Acknowledgements

The NOMAD project was begun by National CSS, Inc., in July, 1973, under the direction of Harold Feinleib, now Vice President of Advanced Technology for the company. Among the initial team members were Judd Boykin, now Director of Software and Applications Products, Nick Pisarro, now a Consulting Programmer in the Advanced Technology group, and Nick Rawlings, now a manager of Advanced NOMAD Usage Projects. Now managed by Doug Grant, NOMAD development continues, as experience has shown ways in which it could be improved and strengthened to meet the needs of more users.

This book was written for National CSS and at the direction of National CSS personnel. Invaluable help in its preparation was given by Nick Rawlings and by Scot Stoney, a NOMAD Field Consultant. The initial direction of the book preparation was guided by Larry Smith, now an independent consultant. I also wish to express my appreciation to Ruth Morley, Rob Vonderhaar, Bill Clarke, and Seppo Lehto. All of these people, and others unfortunately too numerous to name, were most generous with their time and expertise. It would have been unthinkable to attempt writing this book without the help of all of them.

The book was written using the time-sharing facilities of National CSS, and employing the SCRIPT system developed by the University of Waterloo. The SCRIPT files containing the text and examples were converted to a form suitable for driving photocomposition equipment by a software product under development by Nick Pisarro. Photocomposition was by Southern New England Typographic Service, Inc., Hamden, Connecticut.

Daniel D. McCracken
Ossining, New York
January, 1980

# Contents

# Chapter 1

# THE APPLICATIONS DEVELOPMENT CHALLENGE

## *Introduction*

This book is intended for the person who wants to get useful work done with a computer, without having to learn data processing. The presentation is based on NOMAD, a software package available from National CSS, Inc., either as part of its time-sharing services or with an NCSS 3200 Series Computer System.

NOMAD is best thought of as a complete application development package. It is based on a sophisticated relational database management system, supported by powerful and flexible commands for loading and modifying database contents and for developing easily-used reports.

## *The problem*

Application development using traditional programming methods is slow, costly, error-prone and difficult to schedule. Programs written in conventional programming languages such as COBOL, Fortran, PL/I and the like, are usually difficult to understand, change and maintain. Really good applications programmers are in short supply.

To some degree, this has been the case since the beginning of the use of computers in business. The problem is becoming more severe, however, as the plummeting cost of computer hardware accelerates the demand for new applications.

In brief, today's information systems manager faces a formidable set of challenges:

1. Increased demand for new applications.
2. Increased complexity of new applications.
3. Increased cost and decreased availability of skilled people.
4. Increased volume of data.

In other words, just when rapidly dropping computer hardware costs are seemingly opening up large areas of new applications, the continuing problems of software development are frustrating the trend. Software costs are on an upward spiral because:

1. The activity is people-intensive.
2. Adequately trained people are in short supply, and training new people is slow and expensive.
3. The tasks are getting tougher.
4. The sheer mass of data is overwhelming old methods.
5. There has never been a real breakthrough in software methodology. Improvements like structured programming and design are significant, but they are not really breakthroughs to be compared with the introduction of transistors or large-scale integration on the hardware side.

## *The challenge*

What we need is a new methodology that:

1. Helps with problem definition.
2. Permits definition of data in terms meaningful to the user.
3. Permits the entry, modification, and deletion of data either interactively or from existing files on traditional storage media.
4. Simplifies the essential operations of applying data integrity constraints and insuring data security.

5. Provides simple ways to specify the reports that are to be derived from the stored data, with useful summaries, flexible formats, selection of data based on screening criteria, relational operators to combine data from different collections, and ways to make only a subset of the data available to users not authorized to see all of it.

It is essential that the new methodology not only permit the easy preparation of standard reports, but also allow the user to specify new reports on an *ad hoc* basis, since it is typically impossible to know at the time of setting up a database all the reports that will be needed from it. This latter requirement is one that traditional procedure-oriented programming serves especially poorly.

## *The NOMAD response*

It is the point of view of this presentation that NOMAD takes data processing a giant stride forward toward these goals. Seen as a complete application development tool, NOMAD provides ease of use, flexiblity, a logical methodology, high reliability, data integrity and security, data independence, and easily-understood commands in which to specify processing. As we shall see, the commands required to specify processing are so much shorter and so much more powerful than the language of traditional procedure-oriented programming, that it does not really seem reasonable to speak of "programming" in NOMAD at all.

In fact, the real breakthrough provided by NOMAD is that for many applications a programmer is not needed. To the extent that this is the case, we have the ultimate solution to the programming bottleneck: eliminate programming.

In a significant fraction of cases, the user himself or herself—after modest training far shorter than required to become a capable programmer—can write requests for new reports, enter and modify data, and design new database applications.

On the other hand, the power of NOMAD is also of great significance to the data processing professional. Applications that would otherwise require weeks or months of concentrated effort are routinely done in NOMAD in days—and lead to much more easily-modified procedures as well. In many cases it is appropriate to use NOMAD to obtain an agreement between the data processing client and the data processing expert as to what is to be done, utilizing NOMAD to develop a prototype of the proposed application. The prototype then replaces the full specifications, which would have taken far longer to prepare, or permit the writing of a more meaningful set of specifications in shorter time.

NOMAD thus offers new power to both the casual user and to the professional, and in so doing gives the two of them an improved means of communication. It therefore offers hope in lessening the gap between data processing departments and users, a gap created in large part by the difficulty in responding to users' needs by means of programming in traditional languages.

## *What this book will do for you*

If you are the person without background in data processing for whom this book is primarily intended, you may reasonably expect that reading it will enable you to use NOMAD by yourself, at least for applications of simple to moderate complexity. In fact, for some simple work, you would not even need to read the entire book, because the order of presentation has been arranged to permit you to get started using NOMAD quickly.

Chapter 2 gives you a sampler of the capabilities of NOMAD for writing reports; Chapter 3 focuses more closely on the same subject. Chapter 4 deals with several features that extend the power of the LIST command: ampersand variables, expressions, and the SET option. Chapter 5 treats data screening and the DEFINE command, and introduces the CREATE command. Chapter 6 brings in the powerful relational operators of NOMAD. Chapter 7 shows the fundamentals of how to enter, modify, and delete the contents of a database, in terms of a case study shown as a sample terminal session. Chapter 8 discusses the rudiments of a NOMAD schema, with which we describe the data to be entered and processed. This chapter also briefly treats the subject of database design in general. Chapter 9 collects the information about accessing, modifying, and verifying database contents, topics that have been presented piecemeal earlier. Chapter 10 similarly collects the material on NOMAD procedures, which make it unnecessary to type long commands at a terminal every time they are needed. Chapter 11 is a second case study, of somewhat more substantial proportions than the examples used earlier, to give some indication of how NOMAD works in larger applications. Chapter 12 gives a sketch of some of the various NOMAD features that it has not been possible to show in detail in an introductory book of this sort. It is assumed that most readers will be familiar with the essentials of using the National CSS facilities, but for those who may lack this background, an appendix presents the basics.

It is worth remarking that the sequence of introduction of topics is essentially the reverse of the order in which things would be done in bringing up a new application. This is done deliberately, for pedagogical reasons and because not every reader will need to know how to implement database systems. The rationale is that many readers will be interested in getting reports from data in a database that already exists, so that material is presented early in the book; even the reader who will eventually be implementing databases himself or herself will presumably be interested in seeing the payoff (easy report preparation) for his or her labors before studying how to bring up brand new applications.

Readers who expect to use NOMAD only to obtain reports from existing databases will find everything they need in the first seven chapters, including, in Chapter 7, a sample terminal session that exhibits all the basic operations.

## *Use the book flexibly*

This book may be used in a variety of ways to meet the needs of readers with varying backgrounds, interests, and goals. Feel free to stop when your immediate needs have been met; you can always read more later. Everything after the next chapter can be read in almost any order that serves your purposes; skip around if you want. By all means run the examples if you have access to appropriate facilities; all of the databases are available by attaching NOMGUIDE on a National CSS system. (The operations for doing this are shown early in the next chapter.) Run the examples as they are shown here, or try your own variations to see what happens.

Good reading!

# Chapter 2

# AN INTRODUCTION TO NOMAD REPORT WRITING

## *Chapter overview*

In this chapter we shall present a sketch of what NOMAD can do for you in writing reports. Using a simple database containing information about a family's stock holdings, we shall see some of the variety of ways in which the data can be displayed as useful information.

The chapter is intended as a first quick overview of the capabilities of NOMAD. In later chapters we shall cover much of the same material in a more detailed and systematic way, and we shall also later cover several important topics that will be omitted in this first look at the subject.

## *The illustrative application: The Jones family stocks*

The Jones family owns stocks in six companies. This number changes, of course, as stocks are bought and sold. The stocks are registered in several different ways: sometimes in the names of the parents (Fred and Martha in the sample data), sometimes in the names of the children (Jane and Fred, Jr.), and sometimes in the names of the two parents jointly. With the various ways of registering the stocks and with purchases having been made on several occasions, the illustrative data describes twenty holdings of the six stocks.

## *Describing the stocks: the schema*

In order to store and retrieve our data, we must describe to NOMAD at least what the items of data are and how we want them printed. In many cases we shall provide somewhat more detail about the data than this minimum, but this much will get us started. The description of the data in the database is called the *schema*. The schema for our illustrative application in this chapter contains two groups of information called *masters*. One master contains basic information about each stock and the other describes each stock holding of the family.

Let us look first at the description of the stocks, as shown on the page opposite, seeing what we can do with that much, ignoring for the moment the actual ownership of stocks by family members.

We see that a schema begins with the word SCHEMA and ends with the word END. All lines are terminated with semicolons. The use of blank lines and most of the details of spacing within the lines are optional and may be used as desired to improve readability.

The line beginning MASTER identifies STOCKMASTER as the name of this *master segment*. The word INSERT has a function that we shall explore in Chapters 7 and 9. The word KEYED informs NOMAD that no two stocks will ever have the same stock code; the system will prevent us from entering duplicate stock codes. It is possible to permit duplicate key values by adding the option NOTUNIQUE, but we we shall have no occasion in this book to utilize this feature.

## *Describing the items*

Next we have descriptions of the five items of data that describe each stock. The names of these data items are of our devising. They can be formed however we wish, using letters, digits, and the underscore, with a maximum of 15 characters, the first of which must be a letter.

The A3 following the item name SYMBOL indicates that the stock code is alphanumeric and contains three characters. The HEADING 'STOCK:CODE' means that when we write a report based on data in this master, the column containing the stock symbol will have the words STOCK and CODE above it on separate lines. The colon between the words STOCK and CODE dictates that the two words are to be on separate lines in the heading.

We see that the name of the stock is 30 alphabetic characters with a heading that will be printed all on one line since there is no colon between the two words.

```
SCHEMA;

MASTER STOCKMASTER INSERT=KEYED(SYMBOL);
   ITEM SYMBOL           A3 HEADING 'STOCK:CODE';
   ITEM STOCKNAME        A30 HEADING 'STOCK NAME';
   ITEM LATESTPRICE      $999.99 HEADING 'LATEST:PRICE';
   ITEM LATEDATE         DATE HEADING 'LATEST:PRICE:DATE';
   ITEM PRICEBASIS       A10 HEADING 'BASIS';

END;
```

LATESTPRICE is a numeric item, as may be deduced from the description of the way we want it printed: with a dollar sign, a maximum of three digits before the decimal point and two digits after the decimal point.

The date of this latest price is specified to be a *date*, which is a special NOMAD data type distinct from either alphabetic or numeric. NOMAD has elaborate and very helpful facilities for entering and printing dates and for carrying out computations on them. To make use of these facilities it is easiest to explicitly designate the item as a DATE data type.

The price basis has to do with the way the prices are reported on different stock exchanges. For the major exchanges we might enter the last price of the day or perhaps the average between the high and low prices for the day; for stock on the over-the-counter market we could enter the bid price or possibly the average of bid and asked; for mutual funds, we might enter the net assets value. There are other possibilities.

To use NOMAD to manipulate data based on this description, three basic steps are required.

1. The schema must be *translated*, that is, converted to an internal form in which it is more readily usable.

2. Data must be entered into the database under control of this description.

3. Commands must be executed to retrieve all or selected portions of the data and present it in a meaningful form.

For now we shall look only at the third step, postponing the other items until later chapters. Here we shall assume that the schema shown earlier has already been translated and used to control the entry of some sample data that will be displayed shortly. After introducing the basic concepts of report writing with a few illustrations based on just the descriptions of the stocks, we shall then exhibit the additional parts of the schema necessary to describe the family's holdings and consider a number of additional examples of what can be done with that.

## *Running the example yourself*

If you have access to NOMAD, either through the National CSS time sharing system or through your own computer, you may wish to run these examples. If you are using the National CSS time sharing services or computing equipment but are unfamiliar with the operating system, you can learn everything you need to know to run these examples from the first few pages of the appendix. Specifically, you need not learn how to use the editor at this time.

```
SCHEMA;

MASTER STOCKMASTER INSERT=KEYED(SYMBOL);
   ITEM SYMBOL            A3 HEADING 'STOCK:CODE';
   ITEM STOCKNAME         A30 HEADING 'STOCK NAME';
   ITEM LATESTPRICE       $999.99 HEADING 'LATEST:PRICE';
   ITEM LATEDATE          DATE HEADING 'LATEST:PRICE:DATE';
   ITEM PRICEBASIS        A10 HEADING 'BASIS';

END;
```

The sequence of operations to run the examples is this:

1. Carry out the sign-on procedure.

2. When the system types the time of day, indicating that you are in the "CSS environment," type the CSS command:

   ```
   attach nomguide
   ```

   The system will respond with a brief message.

3. When the time of day is given again, indicating that you are still in the CSS environment, type the CSS command:

   ```
   nomad
   ```

   The system will respond with an indication of which version of NOMAD you are using. (NOMAD, like most software products, is under continual revision and upgrading.)

4. You are now in the NOMAD environment. When NOMAD is ready to accept a command, it will print the greater-than sign (>). Enter the command:

   ```
   database jonestk1
   ```

   An acceptable abbreviation would be to type simply:

   ```
   da jonestk1
   ```

   Either way, the last character is the digit 1, not the letter l! In many type fonts the two characters are difficult to distinguish. One of the minor skills of a data processor is recognizing such traps, of which another common example is the difference between the digit 0 and the letter O.

   We shall make little use of abbreviations in this book, to promote clarity, but you are free to do so.

   NOMAD will again respond with a greater-than sign. (Whenever you have any doubt about which environment you are in—which happens to everyone from time to time—simply press the carriage return as the first entry on the line; the environment will be explicitly defined for you.)