UNIX® Database Management Systems

UNIX® Database Management Systems

Ulka Rodgers

YOURDON PRESS
Prentice Hall Building
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

RODGERS, ULKA (date)
UNIX database management systems / Ulka Rodgers.
p. cm.
Bibliography: p.
Includes index.
ISBN 0-13-945593-0
1. Data base management. 2. UNIX (Computer operating system)
I. Title. II. Title: UNIX data base management systems.
QA76.9.D3R65 1990
005.75'6—dc 19 89-30740
CIP

Cover design: *Lundgren Graphics, Ltd.*
Cover photo credit: *Slide Graphics of New England, Inc.*
Manufacturing buyer: *Mary Ann Gloriande*

A Division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

Printed in the United States of America

10 9 8 7 6 5 4 3

ISBN 0-13-945593-0

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 ASIA PTE. LTD., *Singapore*
EDITORA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*

Selected titles from the YOURDON PRESS COMPUTING SERIES
Ed Yourdon, *Advisor*

BAUDIN Manufacturing Systems Analysis with Application to Production Scheduling
BELLIN AND SUCHMAN Structured Systems Development Manual
BLOCK The Politics of Projects
BODDIE Crunch Mode: Building Effective Systems on a Tight Schedule
BOULDIN Agents of Change: Managing the Introduction of Automated Tools
BRILL Building Controls into Structured Systems
BRILL Techniques of EDP Project Management: A Book of Readings
CHANG Principles of Visual Programming Systems
COAD AND YOURDON Object-Oriented Analysis
CONNELL AND SHAFER Structured Rapid Prototyping: An Evolutionary Approach to Software Development
CONSTANTINE AND YOURDON Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design
DEMARCO Concise Notes on Software Engineering
DEMARCO Controlling Software Projects: Management, Measurement, and Estimates
DEMARCO Structured Analysis and System Specification
DESALVO AND LIEBOWITZ Managing Artifical Intelligence and Expert Systems
DICKINSON Developing Structured Systems: A Methodology Using Structured Techniques
FLAVIN Fundamental Concepts in Information Modeling
FOLLMAN Business Applications with Microcomputers: A Guidebook for Building Your Own System
FRANTZEN AND McEVOY A Game Plan for Systems Development: Strategy and Steps for Designing Your Own System
INMON Information Engineering for the Practioner: Putting Theory into Practice
KELLER Expert Systems Technology: Development and Application
KELLER The Practice of Structured Analysis: Exploding Myths
KING Creating Effective Software: Computer Program Design Using the Jackson Method
KING Current Practices in Software Development: A Guide to Successful Systems
LIEBOWITZ AND DESALVO Structuring Expert Systems: Domain, Design, and Development
MARTIN Transaction Processing Facility: A Guide for Application Programmers
McMENAMIN AND PALMER Essential System Analysis
ORR Structured Systems Development
PAGE-JONES Practical Guide to Structured Systems Design, 2/E
PETERS Software Design: Methods and Techniques
RIPPS An Implementation Guide to Real-Time Programming
RODGERS UNIX Database Management Systems
RUHL The Programmer's Survival Guide: Career Strategies for Computer Professionals
SCHMITT The OS/2 Programming Environment
SCHLAER AND MELLOR Object-Oriented Systems Analysis: Modeling the World in Data
THOMSETT People and Project Management
TOIGO Disaster Recovery Planning: Managing Risk and Catastrophe in Information Systems
VESELY Strategic Data Management: The Key to Corporate Competitiveness
WARD Systems Development Without Pain: A User's Guide to Modeling Organizational Patterns
WARD AND MELLOR Structured Development for Real-Time Systems, Volumes I, II, and III
WEAVER Using the Structured Techniques: A Case Study
WEINBERG Structured Analysis
YOURDON Classics in Software Engineering
YOURDON Managing the Structured Techniques, 4/E
YOURDON Managing the System Life Cycle, 2/E
YOURDON Modern Structured Analysis
YOURDON Structured Walkthroughs, 4/E
YOURDON Techniques of Program Structure and Design
YOURDON Writing of the Revolution: Selected Readings on Software Engineering

Preface

UNIX has made its debut into the Data Processing world in recent years; its impact is noticeable with the growing number of applications and software packages now available in the marketplace. Custom applications and packaged systems can now be developed using a wide variety of tools, in particular Database Management Systems (DBMS).

The users of any UNIX DBMS need to understand the influence of the operating system on their applications. We also need to know which limitations the implementation of the selected DBMS imposes. Application designers and developers need to understand the tradeoffs involved in selecting one DBMS over another. There is little literature to date that clarifies these issues. This book is an attempt to fulfill this need.

I assume the reader is already interested or involved in the UNIX world, either as a user or developer of applications. This book does not contain introductory material on the UNIX operating system. It assumes you already have some knowledge of data processing and UNIX concepts. It should be useful reading material for

- Users of an application based on a UNIX DBMS who have a technical interest.
- Designers and developers of an application who are either already using or planning to use a UNIX DBMS.
- Data processing managers who wish to gain some background in the subject.
- Readers who wish to increase their knowledge in this area and who may have some knowledge of other environments such as MS-DOS.

This book is a practical guide to the UNIX DBMS world. It is based largely on my experience in selecting DBMSs, implementing and supporting applications based on DBMSs under UNIX. It does not, therefore, reiterate the well-known theories of databases, nor is it a substitute for the reference manuals available with each product.

It covers the necessary foundation for a discussion of tradeoffs in implementing an application. It also discusses the issues faced by the DBMS user community in the UNIX environment. Where a comparison is possible, the differences between MS-DOS, OS/2, and UNIX environments are distinguished, so that the reader can see why DBMSs in each environment are different.

The book is divided into five major parts. Each part consists of several chapters which expand on a common theme.

Part 1: The Theoretical Foundation

This part is a historical perspective on the theories of database management systems. The emphasis is on the relational approach, since the majority of UNIX DBMSs are based on the relational model. We lay the groundwork in this part with the buzzwords common in this industry so you can follow later, more technical discussions.

Part 2: UNIX and DBMS Applications

Chapters in this part examine how database management systems interact with the underlying operating system. UNIX has an influence on what facilities a DBMS can provide and how they are implemented. It also affects the way a developer builds an application. These chapters should be of great interest to application designers, developers, and end-users. Development managers should like some of the project planning tips included in these chapters.

Part 3: Four UNIX DBMSs

All readers should find interesting material in this part. We review the facilities provided by four of the major DBMS products: INFORMIX, INGRES, ORACLE, and UNIFY. We review these products on the basis of the ground rules established in Part 1 and the application needs described in Part 2.

Part 4: Selecting a UNIX DBMS

This part examines the nebulous area of requirements analysis. It contains a lot of practical hints on how to go about choosing a product that best meets your application's needs in its three chapters. A large number of features does not mean that a DBMS will fulfill the needs of every application. This part focuses on how to determine which of the DBMS facilities are important to specific applications.

Part 5: Future Directions

A look at the future developments in the DBMS front. These developments will have a significant impact on the way future applications will operate.

My intention is not to cover all of the theoretical aspects of DBMS. There are already several publications covering this subject well. I simply survey the necessary theoretical concepts to discuss the four DBMS packages described in this book.

Some aspects of the UNIX environment are important in order to understand the options for implementing an application based on a UNIX DBMS. These aspects are discussed in this book. However, I do not intend to describe all aspects of the operating system and its utilities in detail. I only discuss those aspects that interact with DBMSs and applications using them.

There are, of course, many packages available under UNIX at present. The concepts described in Parts 1, 2, and 4 apply equally to these products. I chose four of the most popular products: INFORMIX, INGRES, ORACLE, and ACCELL (UNIFY) for detailed review in the book for several reasons.

Invariably, all four products support the industry standard SQL language. But, some are better suited for small to medium database sizes while others are better for large databases.

Informix has been a widely used product on small UNIX systems: its PC-like user interface techniques show great promise in improving the traditional UNIX methods. INGRES has good productivity improvement tools: it certainly has the widest selection of tools for developers; some of which could almost be used by nontechnical users. ORACLE and ACCELL both use the unusual ***raw disk*** mechanism of UNIX: necessary for producing good performance with multivolume databases. The difference is that ACCELL is native to UNIX, while ORACLE originated elsewhere and was ported to UNIX.

You may question how useful this review of specific versions of products might be. Don't worry, the book discusses the fundamental facilities of these products, not their specifics. Changing these underlying philosophies is more difficult for DBMS vendors than converting your application from one product to another. So vendors are unlikely to make major changes to their base facilities over the next several years. However, a few specific features we discuss might change: So we list the version of the product to which they apply. This book provides sufficient grounding in the basics to enable you to make your own comparisons.

It is almost impossible to discuss every aspect of every system in a single book. This book attempts to cover most of the important features. I encourage you to refer to the manuals for each product for details not covered in this book.

My heartfelt thanks to all of the people who have helped me write this book. My particular thanks to my husband, Paul, who spent countless evenings discussing the ideas and criticizing the contents of this book. Also, thanks to Martin Heneck, Gerry Boyd, and Dennis Pierson whose practical suggestions have helped to make this book useful. Thanks also to R.S. Tare for encouraging me to start this project. My special thanks to the vendors of the products described in this book, for permission to discuss their respective products, for supplying me with the information needed, and clarifying many questions. Finally, thanks to my editor, Ed Moura, and everyone at Prentice Hall who made the production of this book possible.

Ulka Rodgers
Annandale, New Jersey

Trademarks

dBASE is a trademark of Ashton-Tate Corporation.

FOCUS is a trademark of Information Builders Inc.

FourGen is a trademark of Fourgen Software Inc.

IBM, VM/CMS, DB2, IMS, SQL, SNA, LU6.2, APPC, OS/2 are registered trademarks of International Business Machines Corporation.

IDMS is a trademark of Cullinet Software, Inc.

INFOEXEC is a trademark of UNISYS Corporation.

INFORMIX is a registered trademark of Informix Software Inc.
File-it!, REPORT/DB2, Informix Datasheet Add-In, C-ISAM, INFORMIX-SQL, RDSQL, INFORMIX-TURBO, INFORMIX-ESQL/C, INFORMIX-ESQL/COBOL, INFORMIX-4GL are trademarks of Informix Software, Inc.

INGRES is a registered trademark of Relational Technology.
Applications-By-Form, INGRES/APPLICATIONS, INGRES/EQUEL, INGRES/ESQL, INGRES/FORMS, INGRES/GRAPHICS, INGRES/MENU, INGRES/NET, INGRES/PCLINK, INGRES/QUERY, INGRES/STAR, INGRES/REPORTS, Query-By-Forms, Report-By-Forms, Visual-Forms-Editor (VIFRED), Visual-Graphics-Editor (VIGRAPH) and Visual Programming are trademarks of Relational Technology.

Lotus 1-2-3 is a trademark of Lotus Development Corporation.

MS-DOS, Microsoft OS/2 is a trademark of Microsoft Corporation.

ORACLE is a registered trademark of Oracle Corporation.
Easy*SQL, SQL*Forms, SQL*Plus, SQL*QMX, SQL*Report, SQL*Report Writer, PRO*C, SQL*Menu, SQL*Net, SQL*Connect, SQL*Star, SQL*Calc, SQL*Loader are trademarks of Oracle Corporation.

PROGRESS is a trademark of Progress Software Inc.

SYBASE is a trademark of Sybase, Inc.

TUXEDO is a trademark of AT&T.

UNIBATCH is a trademark of Unisystems Software Ltd.

UNIFY is a registered trademark of Unify Corporation.
ACCELL, Direct HLI, ENTER, PAINT, RPT are trademarks of Unify Corporation.

UNIX is a trademark of AT&T Bell Laboratories.

UQUEUE is a trademark of Unitech Software Inc.

Contents

Part 1

The Theoretical Foundation

Despite the title, this part is actually only an overview of the theories underlying DBMS technology. It aims to introduce you to the terminology prevalent in this industry, rather than to provide a rigorous discourse on the technicalities. For this reason, we merely skim the surface of the theories in an informal manner.

One of the primary objectives of the following chapters is to provide sufficient background so you can follow the later, more technical parts of the book. We do, however, cover some very important ground on what DBMS products do for your development methods and why you would want to use them. Chapter 1 discusses some of the problems developers face when using non-DBMS data storage methods.

Chapter 2 covers the basic terminology used in the relational database management systems and some of the useful data design techniques. The remaining two chapters concentrate on describing the types of facilities found in commercial products. The purpose is to establish some ground rules on how UNIX affects these facilities, which facilities are offered by the products discussed in this book, and probable directions for future developments.

1

What Is a DBMS?

Everyone *knows* what a database management system is, right? But what is the difference between a ***database*** and a ***database management system*** (DBMS)? We have all heard these buzzwords for quite some time; some of us even use them every day. We don't always know what differentiates them from each other. That is what this chapter is all about.

We hear that we must use these DBMS products, that they are the way of the future. But it is rather difficult to judge what they really do for you. Over the next few sections, we will take a look at where these ideas came from and some of the buzzwords common in the industry. We will also discuss what these products mean to your company no matter which business you are in.

Although the book itself is primarily aimed at the UNIX environment, most of the discussions in this chapter do not relate to any specific operating system. After all, UNIX was not commercially available until after these principles had matured.

1.1 A Historical Perspective

We have had databases ever since man started keeping written records. Whether carved into stone slabs, written on paper and stored in a file cabinet, or stored electronically in a computer, a database is simply a store of data. Of course, in the early days of computing and even today, we used the term ***files***. Because of the limitations of the early computer hardware, most of these files were stored on cards and paper tapes. With the advent of magnetic storage media, we started using magnetic tapes, and finally keeping these files on-line on disks.

The buzzword database appeared at about the time we started keeping more and more of our files on-line. It refers to a collection of files that are related in some way. For example, we speak of the accounting database, the marketing database, or a personnel database. Each database might consist of several files such as general ledger, vendor, and customer files in an accounting database.

So, if we have had databases for such a long time and managed to use them to our advantage, why do we need a DBMS? The answer lies in how changes in our methods of software development over the years have improved our development productivity. We need to examine how systems are built without a DBMS and why it helps to improve such systems development.

The earliest software was written in machine code using the basic computer terms of 1's and 0's. Then, we developed ***assembler*** languages to make it easier to write this software. We call the era of machine code programs as ***first generation*** languages, and assembler as ***second generation*** languages. With assembler languages we could write software and make it work much more quickly than we could when writing in machine code. In other words, we increased productivity of the software developers.

Then came high level languages such as COBOL, FORTRAN, and later many others. We call these the ***third generation*** of programming languages. Again, we increased development productivity by an order of magnitude because these languages were even easier to program in than assembler. Besides, many more people could learn and effectively use these languages than those who could make sense out of assembler languages. A DBMS improves the development productivity yet again by taking these developments to the next logical step. Its ***fourth generation*** languages provide a higher level of development interface than the third generation programming languages like COBOL.

A DBMS provides many tools to speedily develop screen interfaces such as menus and interactive forms which require several pages of third generation programming language code. By providing standard tools, they not only reduce repetitive code in programs but also enforce consistency in the user interface. A DBMS offers you an easier and faster way of developing typical forms-based interactive programs. Similarly, it offers report generators, so you can produce a typical report with a few commands, not pages and pages of code in a language like COBOL. These tools improve productivity in the same way as languages like COBOL improved it over assembler languages: by giving you a higher level of development interface.

A DBMS changes your views on accessing data with its end-user query tools. Non-technical people could use such ad hoc query tools with a little training. You still need programmers to write programs for prettily laid out reports or those that are used frequently. But, if you need the data *now*, and are not fussy about the layout, ad hoc query tools let you do the job yourself!

While there were improvements in development languages, operating systems also improved to relieve us from developing code to access storage and peripheral devices. Operating systems manage hardware devices and the running of programs. In the context of data management, what these improvements meant was that we no longer had to worry about which disk block a particular file started and ended. We could refer to files by a meaningful name, and specify a record in the file as being a specific size. The file management portion of the operating system translates these file names and record specifications into blocks on disks.