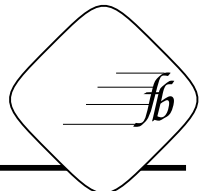




FirstBase

firstbase software
7090 N. Oracle #178-165 • Tucson, Arizona 85704
(520) 742-7897 • firstbase@firstbase.com



'The horror of that moment,' the King went on, 'I shall never, never forget!' 'You will, though, the Queen said, 'if you don't make a memorandum of it.'

Through The Looking Glass
LEWIS CARROL

*There are no such things as applied sciences,
only applications of science.*

LOUIS PASTEUR

Give us the tools and we will finish the job.

WINSTON CHURCHILL

This is just computer science; there is nothing new.

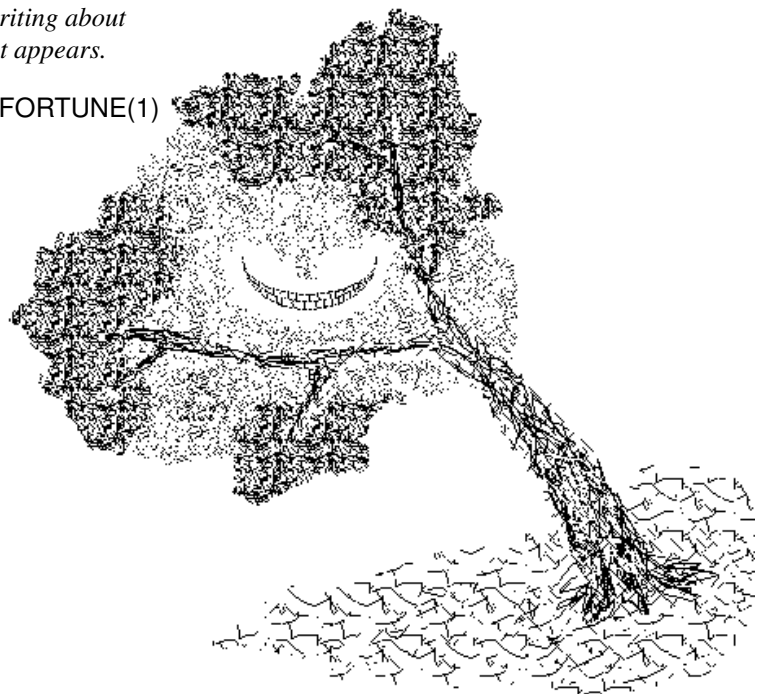
DAVID HANSON

If all you have is a hammer, everything looks like a nail.

Baruch's Observation

Has everyone noticed that all the letters of the word "database" are typed with the left hand? Now, the layout of the QWERTYUIOP typewriter keyboard was designed, among other things, to facilitate the even use of both hands. It follows, therefore, that writing about databases is not only unnatural, but a lot harder than it appears.

FORTUNE(1)



Credits and Acknowledgments

FirstBase is the proprietary database management system from FirstBase Software. All tools were conceived of and written by FirstBase.

A special thanks to all those that made this toolkit a reality — shibumi, slice, pj, the e.t.s. who is getting better, epo, chani, the fac, ime, ode, xted, and the hidden cast, and the vamps, and the hidden cast.

Trademarks

UNIX is a trademark of Bell Laboratories

Copyright 1990 – 1996 by FirstBase Software

Revision VII-H

This publication is protected by the Federal Copyright Law, with all rights reserved. No part of this publication may be reproduced, stored in a retrieval system, translated, transcribed, or transmitted, in any form, or by any means manual, electric, electronic, electromagnetic, mechanical, chemical, optical, or otherwise, without prior written permission from FirstBase Software.

The *FirstBase* source code is a confidential trade secret of FirstBase Software. You may not attempt to decipher or decompile *FirstBase* or develop source code for *FirstBase*, or knowingly allow others to do so. *FirstBase* and its documentation may not be sublicensed and may not be transferred without the prior written consent of FirstBase Software.

Preface

The *FirstBase User's Guide* provides comprehensive documentation on the operation of the tools in *FirstBase*.

The first part of this manual is the tutorial section of the User's Guide. These chapters contain all the information needed to create useful, every day database applications using *FirstBase*. It is not necessary to understand the computer's underlying operating system, UNIX, to use *FirstBase*.

The second half of this manual contains standard UNIX style manual (*man*) pages. These are the reference sections of the documentation, and cover each *FirstBase* tool in full detail.

firstbase software

7090 N. Oracle #178-165 • Tucson, Arizona 85704
(520) 742-7897 • firstbase@firstbase.com

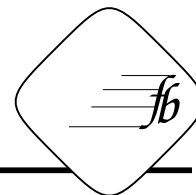


Table of Contents

An Introduction to FirstBase 1-1

FirstBase and Manual Overview 1-1

Types of Tools in FirstBase 1-1

Dictionary Tools 1-1

Data Tools 1-2

Chapters of the User's Guide 1-2

An Introduction to FirstBase 1-2

Communicating with FirstBase 1-2

Operating FirstBase 1-2

Defining a FirstBase Database 1-2

The Database Editor 1-2

Creating Custom Screens and Views 1-3

Creating FirstBase Reports 1-3

Creating an Index 1-3

Creating Mailing Labels 1-3

Document Merging 1-3

Removing/Restoring Deleted Records 1-4

Reshaping a Database 1-4

Database Concatenation 1-4

Database Joining 1-4

Global Database Updates 1-4

Downloading/Uploading Databases 1-4

Creating FirstBase Menus 1-4

Manager Tasks 1-4

Advanced FirstBase Topics 1-4

Communicating with FirstBase 2-1

The Computer Terminal 2-1

The Monitor 2-1

The Keyboard 2-1
The <SHIFT> Key 2-1
The <CONTROL> Key 2-2
The <BACKSPACE> Key 2-2
The <RETURN> Key 2-2
Numeric Keypad 2-3
Function Keys 2-3
Connecting Your Terminal to the Computer 2-3
 Dialing Through the Modem 2-3
 Logging Onto The Computer 2-3
 Logging Off The Computer 2-4
Formats of Communicating with FirstBase 2-4
 Entering Data - The Input Dots 2-4
 Correcting Data 2-5
 Data Entry Errors 2-5
 The END key - '-' 2-6
 The Help Key - <CTL>-H 2-7
 The FirstBase Screen 2-7
 The Screen Header 2-7
 The Screen Footer 2-8
 The Command Line 2-8
 FirstBase Errors 2-9
 Recoverable Errors 2-9
 Fatal Errors 2-10
 Common FirstBase Screens 2-10
 Any Change Screen 2-10
 Yes/No Questions 2-11
 FirstBase Choose Field Mechanism 2-12
Summary 2-12

Operating FirstBase 3-1

Getting Started 3-1

The FirstBase Menu Shell	3-1
FirstBase MAIN Menu	3-1
Using FirstBase Menus	3-2
Exiting the Menus	3-2
Making a Menu Selection	3-3
Environment Commands	3-3
Environment Detail	3-4
FirstBase File Names	3-5
What's in a Name	3-5
Changing a File Name	3-5
Change Working Directory	3-5
List FirstBase Files	3-6
External UNIX Shell	3-6
UNIX Command	3-6
Summary	3-6

Defining a FirstBase Database 4-1

What is a Database	4-1
Getting Started	4-2
Walk Through	4-2
Starting the Define Database Tool	4-2
The Define Database Screen	4-3
The First Field Definition	4-4
Finishing the First Field	4-5
Complete the Database Definition	4-6
Correcting Errors	4-7
AutoIndex the Database	4-8
Exiting the Define Database Tool	4-9
Some Features	4-9
More on Database Creation	4-9
Words of Encouragement	4-10
Database Considerations	4-10

Field Typing 4-11

 Alphanumeric vs. Numeric 4-11

The Field Default 4-12

Summary 4-13

The Database Editor 5-1

Concepts Behind Dbedit 5-1

The Database Revisited 5-1

Overview of Dbedit 5-2

Getting Started 5-2

The Beginning: The Dbedit Command Screen 5-3

 Now What? 5-4

 A Sample Command Screen 5-4

 The Field Lines 5-4

 The Cursor 5-5

Dbedit: Overview of Control Levels 5-5

Dbedit: Command Level Mode 5-5

 Adding New Records - '@' 5-6

 Locating/Modifying A Record - <key> 5-7

 Exiting the database editor - '-' 5-9

 More Simple Commands 5-9

 More Record Location Commands 5-10

 Regular Expression Record Searching 5-10

 Searching on Non-Indexed Fields 5-11

Dbedit: Record Level Mode 5-11

 Viewing a Record: <RETURN>, 'b', # 5-11

 Exiting Record Level: END, <CTL>-X 5-12

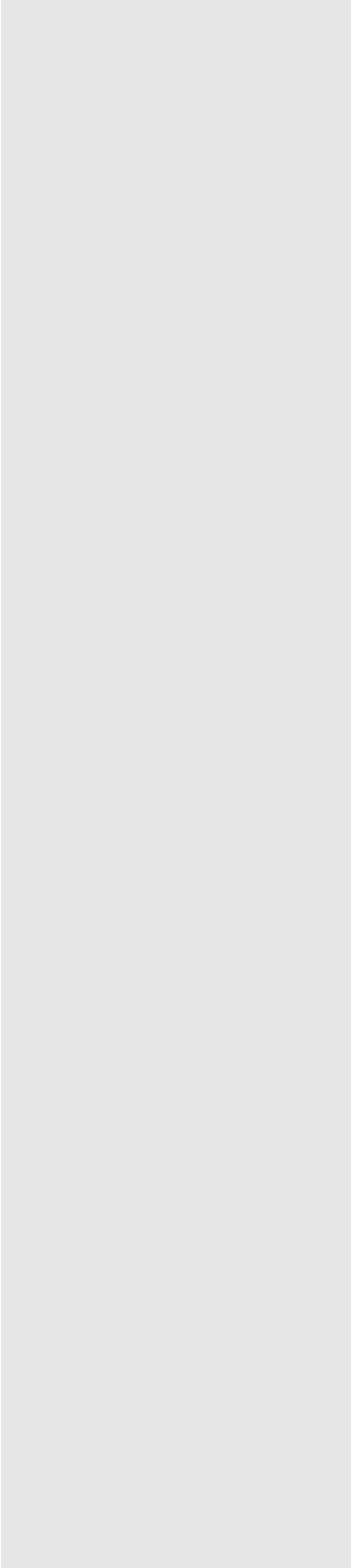
 Deleting The Record: 'del' 5-12

Dbedit: Field Level Mode 5-13

 Entering And Exiting Field Level 5-13

 Editing The Field 5-15

 Alphanumeric - a 5-15



Alpha - A 5-15
Binary - b 5-15
Choice - c 5-15
Silent Choice - C 5-16
Date - d 5-16
Dollar - \$ 5-16
Floating Point - f 5-16
Formula - F 5-16
Link - L 5-17
Numeric - n 5-17
Positive Numeric - N 5-17
Upper Case Alphanumeric - U 5-17
Extended Choice/Help - X 5-17
Walk Through Data Entry 5-18
 Adding A Record 5-19
 Corrections 5-19
 Saving Your Record 5-19
 Exiting Add Mode 5-20
Summary 5-20

Creating Custom Screens and Views 6-1

Database Screens 6-1
The Screen Dictionary 6-2
Getting Started 6-2
 The 'Define Screen' Screen 6-3
 The Field Column 6-3
 The Default Column 6-3
 Making Corrections 6-3
 Exiting the Define Screen Tool 6-4
Database Views 6-4
Summary 6-5

Creating FirstBase Reports 7-1

Defining a Report 7-1

 Getting Started 7-1

 Creating the printout definition 7-2

 The Field Column 7-2

 The Easiest Method 7-2

 Field Totals and Breaks 7-3

 Corrections and Features 7-3

 Miscellaneous Report Information 7-4

Generating a Report 7-5

 Automatic Index Regeneration 7-5

 Interactive Report Generation 7-5

Viewing and Printing an Existing Report 7-7

More Complex FirstBase Reports 7-8

Summary 7-8

Creating an Index 8-1

What is an Index 8-1

Getting Started 8-1

Creating the Index Definition 8-2

 Starting the Define Index Tool 8-2

 The Define Index Screen 8-3

 A Selection Criterion 8-3

 The Field Column 8-3

 The Val1 Column 8-3

 Finishing One Entry 8-4

 The 'Sort By' Screen 8-5

Generating an Index 8-7

Another Example 8-8

 The Val2 Column 8-9

 Logical Connectors - And/Or Column 8-10

 Completing the Example 8-11

A Word on AutoIndexes 8-11

Index Internals 8-11

Normal Indexes 8-11

Btree Indexes 8-12

Summary 8-13

Creating Mailing Labels 9-1

Mailing Labels 9-1

Getting Started 9-1

Creating the Label Definition 9-1

Starting the Define Label Tool 9-1

The Define Label Screen 9-2

The Label Template 9-3

Miscellaneous Label Information 9-4

Making Corrections 9-5

Generating Labels 9-6

Automatic Index Regeneration 9-6

Interactive Generation 9-6

Viewing and Printing Existing Labels 9-7

Summary 9-7

Document Merging 10-1

Merging Databases with Text Files 10-1

Getting Started 10-2

Creating the Merge Source 10-2

Running the Document Merge Tool 10-3

Automatic Index Regeneration 10-3

Interactive Generation 10-3

Viewing and Printing Merged Document 10-4

Summary 10-4

Removing/Restoring Deleted Records 11-1

Deleted Database Records 11-1

Getting Started 11-1

Packing and Undoing 11-2

Summary 11-2

Reshaping a Database 12-1

Converting a Database 12-1

The Conversion Dictionary 12-1

Uses of Database Conversions 12-2

Getting Started 12-3

Creating the Conversion Definition 12-4

 The Define Conversion Screen 12-4

 The Field Column 12-4

 The Type Column 12-5

 The Size Column 12-5

 The Conversion Status 12-5

 Miscellaneous Conversion Information 12-5

 Making Corrections 12-5

Generating a Converted Database 12-6

 Automatic Index Regeneration 12-7

 Interactive Generation 12-7

 Database Checking 12-8

Other Database Conversion Methods 12-8

Summary 12-8

Database Concatenation 13-1

Connecting Databases 13-1

Running The Concatenation Tool 13-1

Summary 13-2

Database Joining 14-1

Joining Two Databases 14-1

Getting Started 14-2

Running The Database Join Tool 14-3

 Automatic Index Regeneration 14-3

 Interactive Generation 14-3

Summary 14-4

Global Database Updates 15-1

Updating a Database 15-1

The Update Dictionary 15-1

Uses of Database Updates 15-1

Getting Started 15-2

Creating the Update Definition 15-2

 The Define Update Screen 15-2

 The Field Column 15-3

 The Update String Column 15-3

 Deleting Records 15-4

 Making Corrections 15-4

Generating Database Updates 15-5

 Automatic Index Regeneration 15-5

 Interactive Generation 15-5

Summary 15-5

Downloading/Uploading Databases 16-1

Emitting Database Values 16-1

Uploading into FirstBase 16-1

Summary 16-2

Creating FirstBase Menus 17-1

- Menu Dictionaries 17-1
- The Screen Section 17-1
- The Actions Section 17-1
- Example 17-2
 - Special Action Flags 17-3
- Summary 17-3

Manager Tasks 18-1

- Database Security 18-1
- Multi User Databases 18-1
- Database Integrity 18-2
- Database Cleaning 18-2
- Database Recovery 18-3
- Environment Control Hints 18-3
- Other Points Of Interest 18-3
- Summary 18-3

Advanced FirstBase Topics 19-1

- EZ FirstBase 19-1
- Application Menus 19-1
- Intraline Editing 19-2
- FirstBase languages 19-2
 - Standard awk Mechanism – dbawk 19-2
 - Macro Language – dbmacro 19-2
 - Structured Query Language – dbsql 19-3
- View Dictionary Hints 19-3
 - Text Fields 19-3
 - Macro Fields 19-3
 - Trigger Fields 19-4
- Applications 19-4

FirstBase Installation A-1

Installing FirstBase A-1

More On Fixed CPU FirstBase Licenses A-3

Local or Global A-3

Support Date A-4

License Type A-4

Host ID A-4

Fixed License Password A-4

Fixed License Completion A-4

More On Floating FirstBase Licenses A-5

Sequence Number A-5

Exclusive User ID A-5

Floating License Completion A-6

Testing The Installation A-6

Internal Notes A-6

Steps for Defining a FirstBase Database B-1

Conversion to FirstBase C-1

Cdb Toolkit to FirstBase C-1

Other FirstBase Modifications C-1

Where shall I begin, please, your Majesty?', he asked. 'Begin at the beginning,' the King said, gravely, 'and go till you come to the end: then stop.'

Alice's Adventure in Wonderland
LEWIS CARROL

One can't proceed from the informal to the formal by formal means.

FORTUNE(1)

A magnitude that is divisible in one dimension only is a line, divisible in two dimensions a plane, in three a body. There are no further possibilities, for three dimensions are all that exist, and to divide in three dimensions is all that can be done. For as the Pythagoreans say, the All and everything in it are determined by the number three, since beginning, middle, and end constitute the basic triad that is the number of the All.

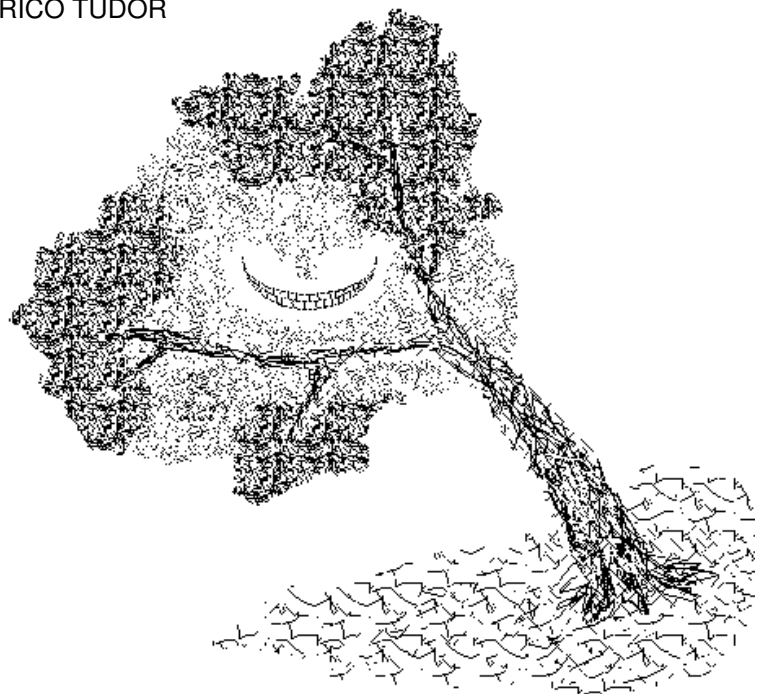
ARISTOTLE

The last thing one knows in constructing a work is what to put first.

BLAISE PASCAL

In the beginning there was data. The data was without form and null, and darkness was upon the face of the console; and the Spirit of IBM was moving over the face of the market. And DEC said, "Let there be registers"; and there were registers. And DEC saw that they carried; and DEC separated the data from the instructions. DEC called the data Stack, and the instructions they called Code. And there was evening and there was morning, one interrupt...

RICO TUDOR



An Introduction to FirstBase

This chapter provides an overview of *FirstBase* and a description of the chapters in this manual.

***FirstBase* and Manual Overview**

FirstBase is a large set of end user programs designed to create, manipulate, organize, and format relational databases and database systems.

These tools are used to produce and maintain information tracking applications and systems of data based on relational databases.

This manual explains in detail how to create and manipulate simple databases and database structures.

This level of documentation attempts to avoid the underlying UNIX operating system and makes no assumptions about the reader's computer or database knowledge. Furthermore, all the *FirstBase* tools are presented as menu selections.

Although this approach is not entirely appropriate for those who know UNIX, it can still serve as a starting point for anyone unfamiliar with *FirstBase*. The detailed, UNIX style manual pages are included as well.

Types of Tools in *FirstBase*

While *FirstBase* contains more than enough tools to develop and maintain relational database systems, all of the tools can be classified into two types.

Dictionary Tools

Dictionary tools from *FirstBase* are used to create and modify the structures of a database and other data objects.

Formal definitions for all data objects, including databases, indexes, and printouts are defined using these dictionary tools. The definitions are stored in files referred to as dictionaries.

All of the *FirstBase* dictionary tools are simplified full screen editors that allow manipulation of data dictionaries in a uniform manner. These dictionaries are created and maintained in the same manner as the actual database will be.

Data Tools

Data tools from *FirstBase* are used to organize and manipulate the data within a database system.

These tools allow viewing and scanning of data, and generate data related objects such as databases, indexes and printouts.

Each of these data tools relies on one or more data dictionaries created by the dictionary tools. Most of these tools process some or all of a database, in either interactive or batch mode.

The generated object is then used in the database system, or as a formatted display or product for people.

Chapters of the User's Guide

The chapters of this manual contain all the information needed to create useful, every day database applications using *FirstBase*.

An Introduction to FirstBase

You are reading it now.

Communicating with FirstBase

Communicating with *FirstBase* covers the beginning of how to talk to a computer. Keyboards, logging in, and terminals are covered in detail. The methods *FirstBase* will use to accept and display information are also described.

Operating FirstBase

This chapter explains in detail how to access the various parts of the *FirstBase* system. *FirstBase* comes complete with a menu shell tool that can be used to operate and control all of the *FirstBase* tools. Display and control over the *FirstBase* environment is also accomplished here.

Defining a FirstBase Database

Defining a database is the first step to creating a database application. For many applications, this is virtually the only step needed to get a useful application up and running. This chapter covers the creation and modification of the database dictionary.

The Database Editor

The database editor is a simplified full screen editor that allows complete maintenance and scanning of databases on the record level. This chapter describes the major points needed to use the database editor, the heart of *FirstBase*.

You will use the database editor to enter and modify all data you want stored in the *FirstBase* system.

Creating Custom Screens and Views

This chapter details how to define a custom screen for a *FirstBase* database. A custom screen for the *FirstBase* system merely rearranges selected fields of a database for the database editor.

Creating FirstBase Reports

Details on creating a report are covered in this chapter. Both the creation of a dictionary and generation of a report are discussed. The standard *FirstBase* report supports multiple subtotals and break values.

Creating an Index

Once data has been entered, an index can be created. Although automatic indexes can be built into the database, often this feature is not enough. This chapter covers how to create sophisticated, highly selective indexes from a database for use in organizing and formatting the database information.

These generated indexes can be used anywhere in the *FirstBase* system to impose an ordered structure on the database. This feature works with the database editor, the printout generator, and most other *FirstBase* tools.

Creating Mailing Labels

This chapter covers all you need to know to define and generate mailing labels. Complete control over the shape and layout of the labels is supported.

Document Merging

Merging a text document with records of a database is covered in this chapter. This requires a bit more UNIX knowledge than any of the other tools, but provides far more power than the standard *FirstBase* report generator.

Document merging is used to plug pieces of database information into a text template that will be recreated for each desired record of the database.

The *FirstBase* document merging mechanism greatly simplifies tasks such as mass mailings, invoices, specialized printouts and data formatting.

Used as a pre-processor to the UNIX text formatting utility, *troff(1)*, the power is increased a thousand fold.

Removing/Restoring Deleted Records

When records are deleted using the database editor, they are actually flagged for deletion. This chapter covers how to physically remove deleted records from a database, and how to logically restore these records as live data.

Reshaping a Database

A database can be split and reshaped without losing any data. This can serve to filter out portions of a database, or add more structure to a database. Reshaping instructions are outlined in this chapter.

Database Concatenation

Databases can be concatenated or pasted together end to end. All the details for accomplishing this task are described in this chapter.

Database Joining

Databases can also be joined or put together side by side. The methods for accomplishing this task are described in this chapter.

Global Database Updates

Sometimes it is necessary to make global changes to information stored in a database. This chapter describes how to selectively update database information for large sections of a database.

Downloading/Uploading Databases

All the needed mechanisms for down-loading data into and uploading data out of a *FirstBase* database are covered in this chapter. The basic comma separated format is covered, with ideas on more sophisticated formats.

Creating FirstBase Menus

This chapter covers the creation of *FirstBase* menus. The design and execution of individual menu components is explained in detail.

Manager Tasks

This chapter covers tasks that are managerial in nature. These tasks include pointers on creating and modifying menus, database security, and database integrity.

Advanced FirstBase Topics

More advanced topics within *FirstBase* are covered here. These issues include more complex uses of the *FirstBase* macro language, intraline editing, view dictionaries, and trigger fields.

*For your own good, turn the pages of your
Greek exemplars by night and by day.*

HORACE

Let your communication be Yea, yea; Nay, nay.

St. Mathew 5:37
BIBLE

*A birdie with a yellow bill
Hopped upon the window-sill,
Cocked his shining eye and said:
'Ain't you 'shamed, you sleepy-head?'*

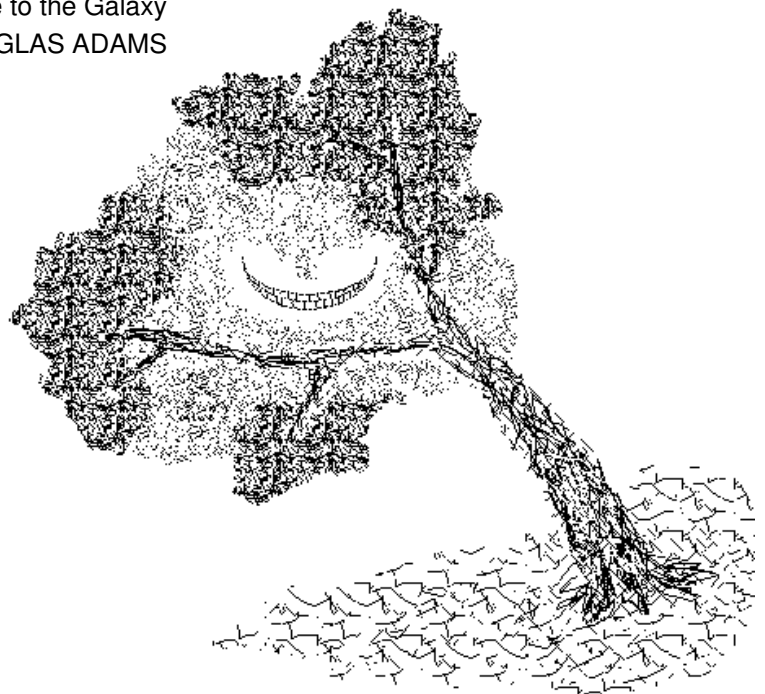
Time to Rise
A Child's Garden of Verses
ROBERT LOUIS STEVENSON

*There is no mistake; there has been no mistake;
and there shall be no mistake.*

DUKE OF WELLINGTON

*"You know, it's at times like this when I'm trapped in a Vogon airlock
with a man from Betelgeuse and about to die of asphyxiation in deep
space that I really wish I'd listened to what my mother told me when I
was young!" "Why, what did she tell you?" "I don't know, I didn't
listen!"*

Hitchhiker's Guide to the Galaxy
DOUGLAS ADAMS



Communicating with *FirstBase*

This chapter describes the major parts of the computer terminal, the hardware tool used to communicate with *FirstBase*, and how to login or sign onto the computer. The format of all communications with *FirstBase* is also covered in detail.

Users familiar with basic computer operations should skip ahead to "Formats of Communicating with FirstBase" on page 2-4.

The Computer Terminal

The computer terminal is the tool used for talking to a computer. It consists of two major parts, the **monitor** and the **keyboard**.

The Monitor

The monitor is the viewing portion of the computer terminal, often simply called the "screen".

The monitor is used by the computer to send information back to you. Also, keys typed on the keyboard are displayed on the monitor. This allows you to watch what information you send to the computer.

When displaying characters on the screen, a small solid or hollow box the size of a single character is used to mark the progress of the output. This box is called the **cursor**, and is sometimes represented as a blinking box or an underline ("_").

The cursor marks the position on the screen where the next information coming from the computer or the keyboard will appear.

The Keyboard

A standard typewriter keyboard is used to enter characters and numerals into *FirstBase*. In all cases (except password entry!) characters typed on the keyboard are displayed almost instantly on the screen.

The <SHIFT> Key

As with any standard typewriter keyboard, there is a <SHIFT> key that is used to distinguish between upper and lower case. For example, to get "A" hold the <SHIFT> key down, and strike the <A> key. Note that on many keyboards, if you continue to hold the <A> key down, the keystroke will be repeated, generating a string of A's.

The <CONTROL> Key

Computer keyboards have another “shift” key called the <CTRL> (or <CONTROL>) key, sometimes denoted as <CTL>. This key is usually labeled “CTRL” or “CONTROL” on the keyboard, and is located above the <SHIFT> key on the left edge of the main rows of keys.

The control key does not provide any special control inside of *FirstBase*. It is used merely to create a new set of characters.

The <CTRL> key is used exactly like the <SHIFT> key. This key creates a third distinct set of characters — different than lower case characters, different than upper case (<SHIFT>ed) characters.

To type <CTRL>-A, hold the <CTRL> key down, and strike the <A> key.

These control characters are sometimes noted with a caret mark (^) in front of the character, as in “^A”, or by one of the longer versions, “<CTRL>-A” or “<CTRL>A”.

This manual and all software will attempt to adhere to the following notation for control-A: <CTL>-A.

Note all of these characters just mentioned which were generated by using the <A> key are unique: lower case “a”, <SHIFT>-A to get capital “A”, and <CONTROL>-A to get <CTL>-A.

The <BACKSPACE> Key

A very useful key on the keyboard is the <BACKSPACE> key which allows you to backup over input and correct mistakes. This key is usually located on the right edge of the main rows of keys, towards the top.

In *FirstBase*, the <RUBOUT>, , or <DELETE> keys also do backspacing.

The <RETURN> Key

Probably the most used key on the entire keyboard is the <RETURN> key, sometimes labeled <RET> or <ENTER>.

Located on the right edge, roughly opposite the <CTL> key, this key is used at the end of all input sequences to actually send the characters typed to the computer.

Before the <RETURN> is struck, you may backspace over characters or words and then retype that part of the input.

When you are satisfied with what you see on the screen, strike the <RETURN> key once, and the computer will read what you have typed.

Numeric Keypad

Many keyboards also have a numeric keypad off to the right side of the main keys. These keys can be used instead of the numerals located on the top row of the main keys.

Often, there are other useful keys on the numeric keypad, including a period, a comma, a minus sign or dash, and a return <RETURN>.

Function Keys

Various other keys are often provided on computer keyboards. Function keys can appear above and to the sides of the familiar main rows of keys. These keys can be used in the *FirstBase* system if desired, but all actions are also available from the more standard keys.

Connecting Your Terminal to the Computer

The terminal is a needed tool, but it does no good unless it is connected to a computer. If you have a computer on site, then the connection is more permanent. Off site computers require a phone call to connect the terminal to the computer.

Dialing Through the Modem

A **modem** (**mod**ulator - **dem**odulator) is a piece of equipment that allows a terminal to connect to a computer over a standard telephone line. The connection is easy to establish, assuming your terminal is already set to communicate in this manner.

Some modems will dial for you if you type the correct dialing sequence at the modem. These are autodial modems. Consult the modem manual for instructions on the use of the modem.

Logging Onto The Computer

Once you have a login message, you are ready to actually log into the computer. To do this, you will need a login name and password. These will be provided to you by the computer system manager.

Proceed by entering your assigned login name, followed by a <RETURN>. **Always** use **lower** case when logging in.

After you have entered a login or user name, you will be prompted for a password. Note that when you type your assigned password, it will **not** be displayed, or echoed, to the screen.

If you typed carefully, and you have a valid user name and password, then the system will let you in. If the computer denied you access, it was probably a typing mistake. Try again. And maybe a third time. If all else fails, find your computer system manager.

Logging Off The Computer

When you are done using the terminal and computer, you need to formally disconnect the terminal from the computer. It is **very** important that you sign off, or logoff, the computer first. To do this use the <END> key to exit *FirstBase*. (See "The END key - '-' on page 2-6).

If exiting *FirstBase* brings you to the UNIX shell prompt, you can then exit the system by using <CTL>-D, or **logout**. If these do not work, then try **exit**.

You are "off" the computer when you have returned to the "login" message previously described.

Formats of Communicating with *FirstBase*

All of the information going into and coming out of *FirstBase* is watched closely to make sure this information, or data, is healthy. You will enter data and commands into *FirstBase* the same way all the time.

All *FirstBase* tools have similar screens, with headers and footers, and all error messages appear in constant screen locations.

Entering Data - The Input Dots

Whenever you need to input information into *FirstBase*, a series of dots will be displayed on the screen, and the cursor will be positioned on the first dot.

For example, you might see the dots and a message like

```
Enter Selection: [.].....
```

The cursor will be placed on the first dot — the first input location.

These dots, representing a fill-in-the-blank entry area, show the maximum number of characters *FirstBase* will allow as input for that input item or request. It is usually acceptable to enter fewer characters than the maximum.

In some places in the *FirstBase* system only one character of input is needed for a particular task. In these cases, the dot appears under the cursor and may be hard to see.

Often, it is also acceptable to enter a single <RETURN> and accept the **default** value *FirstBase* will plug into the space for you. In some cases, the value inserted can be controlled by each user or each database.

As mentioned earlier, a <RETURN> is required at the end of a series of characters to actually make *FirstBase* read the characters. Until this <RETURN> is typed, you may correct and re-type the data as much as you want.

A few special keystrokes are provided by *FirstBase* to make the entering and correction of data easier.

FirstBase actually has two kinds of input mode, *basic* and *editable*. Although both kinds behave similarly, the *editable* input mode allows user defined keystroke bindings, as well as intraline editing of existing fields of data.

The discussion here is limited to the *basic* input mode. For more information see "Intraline Editing" on page 19-2.

Correcting Data

making
corrections

Backspacing backs up over a recently typed character, erasing that character. The dot that was there is once again shown, and input into the *FirstBase* system can resume.

There are many keystrokes that backspace in the *FirstBase* system, all of them equivalent: <RUBOUT>, , the left pointing arrow (<-), and <BACKSPACE>.

You can also backspace over an entire word with a single keystroke in *FirstBase*. This keystroke is <CTL>-W. (Remember - hold the <CTL> key down and strike the <w> key once. Use the <CTL> key just like you use the <SHIFT> key).

Typing <CTL>-W behaves exactly like typing a series of backspaces. Words are considered to be groups of characters surrounded by white space (tabs or spaces).

It is also possible to backspace over the entire current entry with a single keystroke in *FirstBase*. This keystroke, <CTL>-U, erases the entire line or entry back to the first dot, exposing all the dots. This allows you to quickly abandon an input entry and start again.

Data Entry Errors

FirstBase checks all entries from the keyboard as they are being typed. This means that *FirstBase* can tell before you are done entering whether the data input so far is acceptable.

For example, if *FirstBase* is expecting you to type a number at it, to indicate some numerical choice, and you type the <x> key instead, *FirstBase* knows that "x" is not a numeral, even before you type the <RETURN>. In this case, *FirstBase* will complain loudly.

There are only a handful of input errors, all which are evident and described on the screen as they occur. The major input errors are typing a non-numeral when a numeral is expected, and exceeding the maximum allowable input length.

When *FirstBase* detects an input error, it flashes the screen or rings a bell, and displays an error message describing the nature of the error.

You must then acknowledge that you are awake and have seen the error message by touching (typing) any key on the keyboard to recover from the error.

After recovery from the error, the error message is deleted, and the cursor is placed back in the same place it was before the error occurred. Input into the *FirstBase* system can then continue.

See "FirstBase Errors" on page 2-9 for a description of *FirstBase* errors.

The END key - ‘-’

The END key is a single keystroke that is often used in the *FirstBase* system to exit from a program or a level of a program.

The END key never needs a <RETURN> typed after it — its effect is immediate. If accepted as an END keystroke, the characters “END” will appear at the beginning of the input request dots.

The actual key to use for the END key is the dash or minus sign key, ‘-’. (‘Dash’ and ‘minus sign’ are the same character).

Sometimes the minus sign key ‘-’ will appear on the same key as the underscore key ‘_’. The minus sign is always the unshifted key if the key is for both underscore and minus characters. Often, there is a minus sign on the numeric keypad as well.

Again, this means you do not shift to get the minus sign — just hit the key a single time to achieve the END signal.

The dash is only valid as an END key if the cursor is on the first dot of the input request. But, sometimes the dash is a valid character to be used at this point.

For example, to enter negative 226, you must type ‘-’ ‘2’ ‘2’ ‘6’. To generate an END signal, type two dashes in a row.

Again, to generate an END keystroke signal from a numeric type of field, use two dashes in a row.

In this manual, the END key always refers to this keystroke. Often the *FirstBase* tools will display a messages such as “-=END”, meaning the ‘-’ key will produce the END keystroke.

The Help Key - <CTL>-H

<CTL>-H

In many places within *FirstBase*, help can be requested by entering a <CTL>-H.

If applicable, this keystroke will freeze the current task and page through a helpfile covering your current location within the *FirstBase* system. When you are done with the helpfile, you are returned to the previous task where you left off.

The *FirstBase* Screen

All tools in *FirstBase* have a common screen layout containing a header and a footer.

The header and footer will be displayed in reverse video if possible, with the area in the middle of the screen used by the tool itself for display and/or entry of data.

The Screen Header

The screen header of the tools in *FirstBase* will appear on the top line of the terminal screen. The header contains three distinct areas: *FirstBase* tool name, user title, and *FirstBase* tool status.

The following is a screen fragment depicting a normal *FirstBase* screen header:

```
FirstBase 6.5: dbshell      FirstBase Software      Status: MAIN
```

In this example, the name of the *FirstBase* tool is **dbshell**, and the user title is **FirstBase Software**. This title is under the control of each individual user, the application, or the system, and can be whatever you want it to be.

The third area of the header line is the status area. This area will contain information relevant to the current status of the *FirstBase* tool that is being used. In this example, the status area is displaying **MAIN**, the name of the menu shell file that **dbshell**, this *FirstBase* tool, is using.

Some programs will also utilize a space on the screen immediately below the status area for extra status information. Often this is when a deeper level within the *FirstBase* tool is being used, and both levels need to be shown on the screen.

The Screen Footer

The screen footer appears on the next to last line of the terminal (line 23 for standard 24 line terminals), and is used to display information about the current working conditions, or environment. This information includes which files are in use, user name and the time of day.

The following fragment represents a standard *FirstBase* screen footer:

```
Files: dbase(58/2)  index(55/58)  [john: /usr/tutorial 08:17 06/09/91]
Enter Selection: [. . . . .]      -=End, <CTL>-E=Environment, <CTL>-H=Help
```

In this example, the current database is **dbase**, and it has fifty-eight records in it, with two of them marked for deletion. The current index is named **index**. It has fifty-eight entries, with fifty-five sorted for ultra fast access.

These numbers listed next to the files in the footer are there only for those that need/want them — if they are confusing, ignore them for the moment. More information regarding the meaning of these numbers can be found in *screens*(5).

The area in the footer line set off with square brackets contains some information that changes more often than other areas of the screen.

The user name is shown here, **john**, along with the time and date. Optionally, the current working directory can appear in this area also. (See "Environment Control Hints" on page 18-3 for more details.)

The Command Line

Many *FirstBase* tools utilize the last line of the screen for entry of commands or requests. This line will be referred to as the command line and appears below the footer line.

In these cases, a relevant message will be displayed on the command line, the input dots, as described above, will be shown, and the cursor will be placed on the first of these dots.

In the screen footer above, the command line says **Enter Selection**. There are fifteen dots, with the cursor placed over the first dot. Again, this means that a maximum of fifteen characters can be entered here.

The command line is also used to display on screen help hints. These hints often indicate a few keystrokes that will get you out of what you are doing or display more extensive help instructions.

Usually, these hints will be displayed in the form of 'keystroke=action', such as '-=END' and '<CTL>-E=Environment'. If there are more than one listed, a comma is used to separate them.

errors

FirstBase Errors

There are two types of errors in the programs from *FirstBase*: recoverable and fatal. Both types will print out a message and pause waiting for any key on the keyboard to be struck.

The recoverable error will then allow the tool to continue on where it left off while the fatal errors will cause instant exit from the *FirstBase* tool.

Recoverable Errors

Of the recoverable errors, most involve data entry errors. Either the wrong number or the wrong type of character was entered. The screen footer example can be used to depict this recoverable error state.

If you were to type sixteen 'x's on the keyboard at the screen depicted by the "The Screen Footer" on page 2-8, you would cause a recoverable error since there are only spaces for fifteen characters. The resulting screen (fragment) would be:

```
input error: too long                ** HIT ANY KEY TO CONTINUE **
Enter Selection: xxxxxxxxxxxxxxxxxx  --End, <CTL>-E=Environment, <CTL>-H=Help
```

When an error occurs, the footer line is erased, and the error is displayed in its place. This line will also be displayed in reverse video.

A message indicating the action that caused the error will occur on the left side of this line. In this case, it is **input error: too long**. This is because too many characters were entered at the prompt.

The end of the error line says **HIT ANY KEY TO CONTINUE:**, with the cursor placed after this message.

Again, during recoverable errors, the cursor may be hard to see on some screens since the input field is a single character wide and is next to the right side edge of the reverse video error message.

To recover from the error, you must hit any key on the keyboard. In this case, it really makes no difference what you enter, as long as it is exactly one keystroke.

After this recovering keystroke is entered, the error message will be erased and the footer line that was previously there will be redisplayed. The cursor will return to where it was when the error was detected, and input can continue.

A few other recoverable errors exist in *FirstBase*. In all cases, the error message on the screen will indicate the problem, and the behavior of the tool will be as described here.

Fatal Errors

The fatal errors that come from *FirstBase* take on a very similar form to the recoverable errors with one major difference — after the recovering keystroke is entered, the tool will exit completely.

These fatal errors usually involve a missing or mismatched data file, a file protection problem, or result from the users request to abort the current task. In most cases, a pertinent error message will appear describing the fatal error.

Another difference between the recoverable and fatal error messages is the location of the displayed error message. Since a fatal error message indicates that the *FirstBase* tool is about to exit abnormally, the footer line that displays relevant environment information is left intact.

Instead of using the footer line as the recoverable errors do, fatal error messages appear on the very last line of the screen, the command line. Again, though, the left side of the line shows what caused the fatal error, the right side shows the ‘Hit any key’ message followed by the cursor.

If any fatal errors occur that contain **only** the message **Fatal Error**, notify your computer system manager.

Common *FirstBase* Screens

Many parts of the *FirstBase* system look very similar to other parts. The two major similarities throughout all of the *FirstBase* system are the screen header and footer. These will appear everywhere.

Here are some other similarities among the tools in the *FirstBase*.

Any Change Screen

One item that appears in many places within *FirstBase* is referred to as the **Any Change** screen.

any change

This means there will be some kind of numbered and labeled display in the middle of the screen, and the command line will be asking if you have any changes to make to the displayed data.

Along with the ‘Any Change’ message, there will be some comments in parenthesis that indicate the types of responses that are allowed.

The following is a fairly typical Any Change prompt:

```
Files: dbase(58/2) index(55/58) [john: /usr/tutorial 08:29 06/09/91]
Any Change (#, -=End, <CTL>-H=HELP) ? [.]...
```

Very often throughout the *FirstBase* system, when these numbered displays occur on the screen, the command line comment will display a pound sign ('#') in it, as in the above example. This means a number can be used, as in '1' or '2'.

In the above example, there are two other possible responses. The dash or minus sign key can be used to create the END keystroke discussed in "The END key - '-'" on page 2-6.

Also, by using <CTL>-H, a help screen concerning all the possible commands will be displayed on the screen. (Remember, hold the <CTL> key down and hit the 'H' key once.)

Yes/No Questions

yes/no

Another common request for input from *FirstBase* takes on the form of a yes or no question. For example, one form appears as:

```
Files: dbase(58/2) index(55/58) [john: /usr/tutorial 08:36 06/09/91]
Any Change (y=Yes, <RETURN>=No): [.]
```

Again, the parenthesis surround possible responses from you for this question. To respond affirmatively, saying that 'Yes!' you do want to make changes, you **must** respond with a **y** (followed by a <RETURN>).

The other possible response says <RETURN>. In this case the <RETURN> response will indicate that No!, you do not want to make changes. An **n** followed by a <RETURN> would do the same thing.

Another way of describing these Yes/No question responses is to say the default answer to this question will be an **n**, indicating a negative response.

The END key will also produce a negative response for these types of prompts.

choose field

***FirstBase* Choose Field Mechanism**

Another very common screen found in many of the database dictionary editors is called the Choose Field screen.

The idea behind this screen display is to supply the user with a list of possible fields that can be chosen at that point during the editing of the dictionary.

For example, the following screen shows one of the many database dictionary tools using the Choose Field mechanism:

```
FirstBase 6.5: dbdind          FirstBase Software          Status: Choose Field

      1> Last Name  a      35
      2> First Name a      20 [ ]
      3> Initial   a       1
      4> Telephone a      15

Files: Phone(3/0) Lname(3/3) [john: /usr/tutorial 05:16 01/16/92]
Enter Field # : ...          -=END, <CTL>-C=Skip
```

At this point, you would enter one of the numbers listed to indicate that field, a <RETURN> to indicate the default, an END ('-') to indicate ending, or a <CTL>-X to skip over this entry.

This mechanism is controlled by the *FirstBase* CHOOSEFIELD environment variable as defined in *setup(5)*.

Summary

There is more than enough information in this chapter for the casual user of *FirstBase*. However, there are many other features that are not covered by this chapter. See the manual pages *firstbase(5)* and *input(5)* for pointers to lots more detail.

Who can control his fate?

Othello
WILLIAM SHAKESPEARE

You see, wire telegraph is a kind of a very, very long cat. You pull his tail in New York and his head is meowing in Los Angeles. Do you understand this? And radio operates exactly the same way: you send signals here, they receive them there. The only difference is that there is no cat.

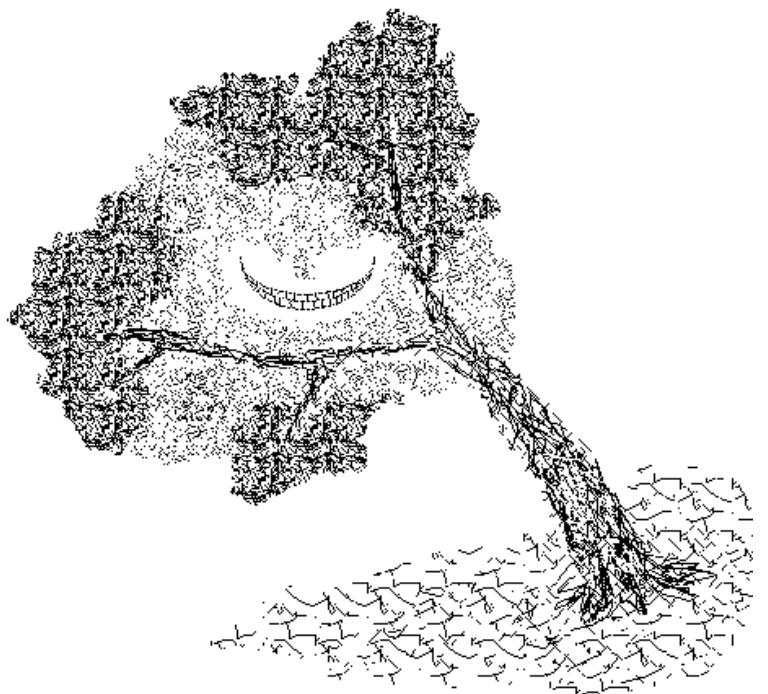
ALBERT EINSTEIN

I claim not to have controlled events, but confess plainly that events have controlled me.

ABRAHAM LINCOLN

Command, n.: Statement presented by a human and accepted by a computer in such a manner as to make the human feel as if he is in control.

FORTUNE(1)



Operating *FirstBase*

This chapter describes how to operate and use the front end menu used to access *FirstBase*.

Although each tool can be run from standard UNIX shells as well, this menu tool provides a framework grouping all of the tools together.

Getting Started

At some point, you will get your terminal connected to the computer, and get past the login message. You may get the actual *FirstBase* system menus, or a UNIX shell command prompt of some kind.

If you get a shell prompt, often a '\$' or a '%', then try typing 'fb' or 'firstbase' and a <RETURN>. This should start up the *FirstBase* menu shell.

If this command fails, see your computer system manager and Appendix A, "FirstBase Installation".

The *FirstBase* Menu Shell

The *FirstBase* system will appear to you as a menu containing a series of menu selections. Each selection consists of either a single word, such as **define**, or a number, like **3**.

To choose a selection, enter the corresponding command followed by a <RETURN>. Only enough of a command to distinguish it from other commands is needed.

When you are ready to log off of the computer, get back to a menu by exiting or finishing whatever you are working on. Then use the <END> key to exit *FirstBase*. (See "Exiting the Menus" on page 3-2).

FirstBase MAIN Menu

Complete control over all of the *FirstBase* tools is available from the MAIN menu. It is also possible to run any of the *FirstBase* tools directly from UNIX. But, this menu tool provides a clean method to use *FirstBase* for most normal database users.

The following screen shows the standard *FirstBase* MAIN menu:

```

FirstBase 6.5: dbshell      FirstBase Software      Status: MAIN
                             FirstBase Main Menu
Database Tools                Sub Menu
define.  Define Database      ezfb.   EZ FirstBase
edit.    Edit a Database      tools.  More Database Tools
scan.    Scan a Database      unix.   Unix command menu
screen.  Define Database Screen secure. Security Tools
vedit.   Visual Edit a Database
vscan.   Visual Scan a Database
query.   Ad Hoc Query

Dictionary Tools              Generator Tools
1. Define Index              7. Generate Index
2. Define Printout           8. Generate Printout
3. Define Labels              9. Generate Labels
4. Define Conversion          10. Generate Conversion
5. Define Update              11. Generate Update
6. Define Merge               12. Generate Merge
Files: dbase(0/0)  index(0/0) [john: /usr/tutorial 14:38 01/16/92]
Enter Selection: .
--End, <CTL>-E=Environment, <CTL>-H=Help

```

Using *FirstBase* Menus

Any menu designed for use with the *FirstBase* menu shell tool will behave in the same manner. The MAIN menu provided with *FirstBase* can be customized.

Other menus can be also created and chained together to form a system of menus.

Each menu selection can activate large sections of commands as illustrated in Chapter 17 *Creating Menus*, and in the manual page for the *FirstBase* menu shell tool, *dbshell*(1).

Exiting the Menus

The first thing to know about using the menu tool is how to quit or exit from the menus. If you are at the *FirstBase* MAIN menu, then the single END keystroke '-' will signal *FirstBase* that you want to exit the menus.

Here you will be given a standard yes/no question asking if you really want to exit the *FirstBase* menu tool. You must respond with a **y** if you want to exit the menu tool.

<CTL>-X

If you are not at the MAIN menu, then you can either use the END keystroke to back out of any sub menus until you are at the main menu, or you can use the abort keystroke, <CTL>-X. (Remember, hold the <CTL> down and strike the 'X' key once). This keystroke will exit any *FirstBase* menu.

Making a Menu Selection

To use the *FirstBase* menus, all you do is choose which selection you want to execute and enter that command.

The provided menu shells in *FirstBase* will display all possible commands in reverse video. Remember, you must follow the command with a <RETURN>.

For example, if want to define a database, you would enter the command **define** (and a <RETURN>). In this case, the *FirstBase* tool for defining a database will be executed.

The selected tool or action will run until completion, then the menu will be redisplayed.

Note that **def** would work here as well, since only enough characters are needed to locate the command.

If an error occurs in the tool that you select, the error message will keep the menu from reappearing until you hit a key on the keyboard. These types of *FirstBase* errors were discussed in "Recoverable Errors" on page 2-9.

The standard help mechanism also works from the command line in the *FirstBase* menu/shell too. This keystroke, <CTL>-H, can be used to display a simple help file.

Environment Commands

The *FirstBase* menu/shell tool is also used to control parts of the environment that each of the *FirstBase* tools will use and refer to.

These items consist of environment settings such as file names and the current working directory.

<CTL>-E

All of the environment commands are available from any of the *FirstBase* menus by using a <CTL>-E keystroke. This command will display the environment screen. (Remember, hold the <CTL> down and strike the 'E' key once).

The result of the <CTL>-E keystroke is the following screen:

```
FirstBase 6.5: dbshell      FirstBase Software      Status: Set Environment

Environment Commands

[d]base      change FirstBase database name
[i]ndex     change FirstBase index name
[s]creen    change FirstBase screen name
[v]iew      change FirstBase view name

[e]nvironment  display current FirstBase environment
[u]nix        unix command
[x]           start unix command shell

[c]hdir      change working directory name
[l]s         list contents of directory in FirstBase format

[p]wd        print working directory
[h]elp       simple help file

<RETURN> or <-> will return to database shell tool

Files: dbase(58/2)  index(55/58)  [john: /usr/tutorial 14:59 01/30/92]
Enter Command: .
```

The most important commands from this environment command screen are discussed in the following sections.

Note that for these environment commands you will **not** enter a <RETURN> to make the computer see the command.

The single keystroke of the first letter of each of the commands will execute that command.

e

Environment Detail

You can display the details of your working environment with the use of the environment command, **e**.

This command will list all file names, whether the files exist or not, the working directory, and other environment variables that control some *FirstBase* features.

FirstBase File Names

The file names are used for naming and locating databases, indexes and screens. The *FirstBase* tools will take care of providing extensions to all of the file names. (Extensions are the part of a file name to the right of a period — for example, ‘idx’ is the extension of ‘index.idx’).

What’s in a Name

tips

The idea is to name files with something that reminds you of their contents — much like a label is placed on a filing cabinet folder. Then you can locate this file easily when you need to.

FirstBase will provide default names to all objects if you do not specify a name. The only problem with this is that these default names do not describe the contents of the files they name. So, it is good practice to provide file names to the *FirstBase* tools.

Note that you can start up the *FirstBase* menu tool with your own set of default file names. See your computer manager and the manual page on *dbshell*(1) for more details.

Changing a File Name

Before changing a file name from the environment command screen it might first be helpful to know what the current file names are.

The database and index names appear on the footer line, as described in "The Screen Footer" on page 2-8. If there is no index, two asterisks are displayed.

To display the current environment settings, use the **e** command.

d

To change a file name, enter the first letter of the object whose name you want to change. For example, enter **d** to change the name of the database. Similarly, an **i** is used to change the index name, an **s** for the screen, and a **v** for the view.

After you enter the name changing command, *FirstBase* will request the name of a file from the command line. Control will then return back to the environment command screen.

Change Working Directory

c

To change the working directory, use the **c** command. This will cause *FirstBase* to request the name of a directory from the command line.

The directory you enter must exist, or *FirstBase* will complain loudly, and not change directories. See your computer system manager if you are unfamiliar with creating a working directory. (Or use the *mkdir*(1) command from the UNIX menu — located below the *FirstBase* MAIN menu).

List *FirstBase* Files

l

The **l** command (lower case 'L') will list all of the files in the current working directory that are *FirstBase* files. This listing is paged to the screen. The *FirstBase* files listed are categorized, sorted and grouped for easy location of file names.

External UNIX Shell

x

The **x** command will begin a UNIX subshell. The shell is determined by the *FirstBase* setup file (see *setup(5)*). The default shell is the Bourne Shell, *sh(1)*. When this shell exits, the environment screen will be redisplayed.

UNIX Command

u

The **u** command is similar to the **x** command, but will first prompt for the command you want to submit to your shell. This could be used to read mail or do some other UNIX task while still inside of the *FirstBase* menu shell tool.

Summary

This chapter has provided a description of the *FirstBase* menu shell tool. This tool is used to group all of the individual *FirstBase* tools into a single unit called the MAIN menu.

If you are interested in all of the details concerning the creation, care and feeding of *FirstBase* menu shell files, see Chapter 17, "Creating *FirstBase* Menus" and the manual page on *dbshell(1)*.

In Architecture as in all other Operative Arts, the end must direct the Operation. The end is to build well. Well building hath three Conditions. Commodity, Firmness, and Delight.

SIR HENRY WOTTON

*When we build,
We first survey the plot, then draw the model;
And when we see the figure of the house,
Then we must rate the cost of the erection;
Which if we find outweighs ability,
What do we then but draw anew the model
In fewer offices, or at least desist
To build at all?*

Henry IV
SHAKESPEARE

If at first you don't succeed, redefine success.

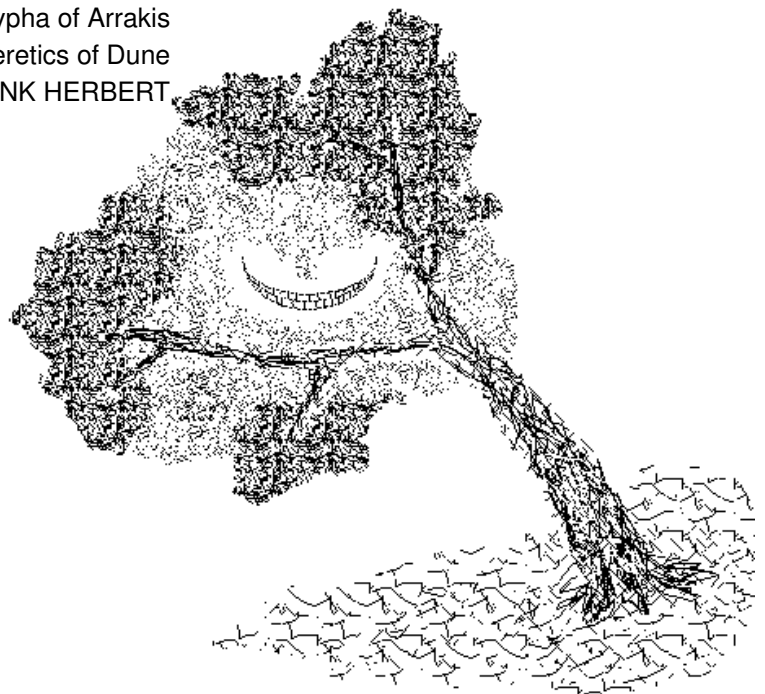
FORTUNE(1)

It is against the grain of modern education to teach children to program. What fun is there in making plans, acquiring discipline in organizing thoughts, devoting attention to detail, and learning to be self-critical?

ALAN PERLIS

Most discipline is hidden discipline, designed not to liberate but to limit. Do not ask Why? Be cautious with How? Why? leads inexorably to paradox. How? traps you in a universe of cause and effect. Both deny the infinite.

The Apocrypha of Arrakis
Heretics of Dune
FRANK HERBERT



Defining a FirstBase Database

The creation and development of a formal database definition is covered in this chapter. For many applications, this is virtually the only step needed to get a useful information system up and running. The *FirstBase* tool covered here is *dbdbas(1)*.

This chapter covers the basics in detail. For a quick list of steps used to create a generalized database dictionary, see Appendix B, "Steps for Defining a FirstBase Database".

What is a Database

A database is an organized package of information. It consists of many smaller packages of data called "records". A record keeps logically connected pieces or fields of information together so this information can be quickly accessed.

For example, if the telephone book is your "database", then a record would be the entire entry for John J. Mills, including phone number and address.

Note that there are many items, or fields, of information in each record in this sample database. Also, each field is for a specific item — last name, first name, middle initial, and phone number.

Formally, a database consists of one or more records, where each record consists of one or more fields

Here is a graphic representation of a simple database:

	Field 1	Field 2	Field 3	Field 4
	Last Name	First Name	Initial	Phone #
Record 1	Smith	Fred	B	299-8771
Record 2	Jackson	Seymour	P	327-8211
Record 3	Mills	John	J	749-2864

Getting Started

Defining a database is nothing more than answering the exact same set of questions about each field or piece of information you want the database to hold. That's it.

You do not need to think of fields in any particular order, or worry about their screen location. The three main questions that you will need to answer for each field are:

name
type
length

- What is this field's Name?
- What type of information will be stored in this field?
- What is the maximum length of this field?

To get started on defining your database think of the major uses of the information you want this database to contain, and try to identify as many of the pieces of information as you can.

Again, note that you will be able to reorder the fields as they appear on the screen at any time using a screen dictionary or a view dictionary, or even physically change their order in the database without losing any stored data you may have already input.

See Chapter 6, "Creating Custom Screens and Views".
See Chapter 12, "Reshaping a Database".

Walk Through

Enough talk. Let's walk through a very simple example and define the simple database shown at the start of this chapter. This simple database will be called the Phone database in this chapter.

Starting the Define Database Tool

The first thing you may want to do is to name your database. This can be done by using the environment command screen and the commands described in "Changing a File Name" on page 3-5.

If you are following the examples from this manual at your terminal, the database name, **Phone**, MUST be entered using this environment command screen. This is a must — otherwise your database will be named something different.

<CTL>-E
d
Phone

Here are the keystrokes if you do not want to look back at Chapter 3: From any of the *FirstBase* menus, enter <CTL>-E, 'd', 'Phone', <RETURN>, <RETURN>. Easy.

Note that **Phone** has one capital letter: 'P'. The rest are lower case. *FirstBase*, and UNIX for that matter, **always** distinguish between upper and lower case.

define

From the *FirstBase* MAIN menu, choose the selection titled **Define Database** to use the *FirstBase* tool for defining a database. I.E., enter the command **define**, followed by a <RETURN>.

The Define Database Screen

The *FirstBase* Define Database screen will look something like:

```

FirstBase 6.5: dbdbas          FirstBase Software          Status: Define Database
Field  Type  Size Default          Comment          C.Loc  Lock?
====  ====  ===  =====          =====          =====
                                          
                                          
                                          
                                          
Files: Phone(0/0)  **  [john: /usr/tutorial 14:54 01/16/92]
Any Change (#, --End, <CTL>-H=HELP) ? ....
```

The idea is to enter one field per line, filling up each of the columns across the top.

**basic
concepts**

The first three columns are the major attributes previously discussed: Name, Type, and Length.

For this first time use, you will only be answering questions for these three columns.

For all other questions and columns, you will use the default provided by the *FirstBase* system and hit a <RETURN>.

All of these keystrokes are described in the following sections.

The First Field Definition

To begin entering field definitions with the database dictionary tool, you will start using what is called *auto add mode*. This means that the editor will accept new additions until you instruct it to stop.

@

To start auto add mode, use the 'at' sign, '@', and a <RETURN>.

After you enter the 'at' sign, the screen will look something like the following picture. This screen shows the first seven characters of the field name already entered. Your display will show only dots where the cursor is.

```

FirstBase 6.5: dbdbas      FirstBase Software      Status: Define Database
  Field  Type  Size Default      Comment      C.Loc  Lock?
  ----  -
1) Last Na[.]..

```

Files: Phone(0/0) ** [john: /usr/tutorial 14:56 01/16/92]
Enter Id, <RET>=Default, -=End

Field

Last Name

You must enter a valid field name here or use the END keystroke to exit back to the command line of the previous screen. Proceed by entering the field name of 'Last Name', and a <RETURN>. This input will be placed under the Field column.

You will have many chances to correct this data if you make a mistake or want to modify it in some way. So do not panic over typographical errors, or erroneous <RETURN> keystrokes.

Type

a

The cursor will then move to the Type column. Note that there is only a single dot here, and it is under the cursor. Enter a lower case **a** (for alpha or alphanumeric) followed by a <RETURN>.

Now, the cursor will move to the column for **Size** and accept an entry for the size of the field you are defining.

Note that the size field only accepts numerals or digits as input — i.e., it wants you to input a valid number.

Size

35

At first this column might look as if it will not line up with its header correctly, but after you hit the <RETURN>, the number you enter will be moved to the right edge of the provided space.

Let's use 35. Enter a '3', a '5' and a <RETURN>.

The fourth column, the default column needs some explanation, and will be explained in detail in "The Field Default" on page 4-12.

Finishing the First Field

For the rest of the columns, hit a <RETURN> and continue on past.

Note that there are more than four more columns, and that the stuff you have entered for this field will disappear for a moment while more questions are asked.

Some columns will even plug values in for you as you <RETURN> on past. These values will be the 'correct' values for the moment.

For example, as you go past the column labeled "default" by hitting a <RETURN>, the characters "*forced entry*" will be inserted in the space on the screen.

Not to worry — these columns can be safely ignored for the moment. And you can always come back later to change things around a bit.

Some of these sub screens you will see will display a piece of information about that field on the line you were on and pause to ask you a question from the command line at the bottom of the screen.

Again, enter <RETURN>s until you get back to the main screen with the Field, Type and Size columns.

7 <RETURN>s

Including the <RETURN> for the default, you will have to enter exactly seven (7) <RETURN> keystrokes to get to the Field column for the next field in our Phone database example.

In fact, for now it might be better if you did not even stop to look at the screens on the way by. The things you see there may do more to confuse you at this point.

Have faith. Hit the 7 <RETURN>s and let's move on.

UNIX
Note

A special note for all the UNIX oriented people: No! this is not the only way to define databases. In fact, this task can sometimes be done much quicker with a text editor.

Use of a text editor or other UNIX tools on any of the standard *FirstBase* dictionaries is permissible and is a feature, but should not be done by the casual user — in any case.

Relevant manual pages are *dictionaries(5)* and *ddict(5)*.

The material presented here discusses the *FirstBase* full screen database dictionary editor, *dbdbas(1)*, which does type and syntax checking on each field attribute.

Complete the Database Definition

finish
entries

To complete the entire database definition for the simple Phone example, you need to define the three remaining fields. (And maybe clean up a few mistakes!).

All of the fields and the values to enter to complete this example are listed in the following table:

	Field Name	Type	Size	Others
Field 1	Last Name	a	35	7 <RETURN>s
Field 2	First Name	a	20	7 <RETURN>s
Field 3	Initial	a	1	7 <RETURN>s
Field 4	Telephone	a	15	7 <RETURN>s

After you have entered all four fields, the cursor will be put on the fifth field awaiting more fields.

But there are no more fields to define in this simple database example.

To exit the *auto add mode* of the define database tool use the END keystroke, '-'.

Correcting Errors

After you exit from *auto add mode*, you should have a screen that looks very similar to the following:

```

FirstBase 6.5: dbdbas          FirstBase Software          Status: Define Database
  Field  Type  Size Default          Comment          C.Loc  Lock?
  ----  -
1) Last Name  a    35 *forced entry*
2) First Name a    20
3) Initial   a     1 *forced entry*
4) Telephone a    15 *forced entry*
                                     no
                                     no
                                     no
                                     no

Files: Phone(0/0) ** [john: /usr/tutorial 14:59 01/16/92]
Any Change (#, -=End, <CTL>-H=HELP) ? ....
```

making
corrections

If your screen does not look like this, then you need to correct it.

Notice that the command line of the screen is waiting for a response. This is the prompt discussed in "Any Change Screen" on page 2-10.

If you enter the number beside the field, you will be able to selectively change any of the items in that field definition.

Once you have started to make a change to an item, the cursor is placed on the **Field** column for the selected field line.

For each piece, or field attribute, the cursor will be placed on that column. If you want to make a change, retype the entry.

If you do not want to make a change to this column, use the abort keystroke, <CTL>-X, to skip to the next column. This keystroke will re-display the value that was there and keep that value intact.

AutoIndex the Database

Now that you are comfortable with the dictionary editing process, we are going to re-edit the first field on the screen, **Last Name**, to make this field automatically create and maintain an index.

Even if you do not understand this concept, blindly follow these next few paragraphs. This one, simple step will save lots of effort and time for those unfamiliar with *FirstBase* indexes.

1

To define an autoindex for the **Last Name** field, enter **1** (one) from the Any Change prompt. The cursor will jump to the Last Name line.

7 <CTL>-X's

Now enter seven (7) <CTL>-X's until you get to the command line prompt and screen that looks like this:

```

FirstBase 6.5: dbdbas      FirstBase Software      Status: Define Database
  Field  Type  Size Default      Comment      C.Loc  Lock?
  =====  =====  =====
  1) AutoIndex File: (VOID)
  2) First Name a      20
  3) Initial   a      1 *forced entry*
  4) Telephone a      15 *forced entry*

Files: Phone(0/0) **      [john: /usr/tutorial] 15:01 01/16/92]
Change Auto Index Info? (y=yes, <other>=no)?

```

y

Respond to the question with a **y** and a <RETURN>.

The cursor will move to the '(VOID)', delete it, and accept input for an index name.

index

To keep things ultra simple, let's name the index **'index'**. So enter 'index' and a <RETURN> (no quotes!).

4 <CTL>-X's

And then hit four more <CTL>-X's, skipping the information requested, to get back to the main Any Change screen.

Exiting the Define Database Tool

When you are done correcting errors for a single field definition, through all columns, the screen will return to base Any Change screen.



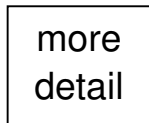
Once you are satisfied with your all of your field definitions, use the END key to exit the database dictionary editor and get back to the *FirstBase* MAIN menu.

If you decide not to keep any of the changes you can abort out of the define database tool using the standard abort keystroke, <CTL>-X. You will be asked if this is really what you want to do via a standard *FirstBase* yes/no question.

If all went well, then you will not need any information in the following sections to complete the Phone walk through example.

Read the summary at the end of this chapter, and then go ahead to Chapter 5 and start entering data into the simple database.

Some Features



If you are creating your own application, playing around, or want to know more, the following sections will provide a bit more detail about the database dictionary editor, and the database dictionary.

There are a few features of the database definition tool that may help you in correcting and organizing your field definitions.

These features are described in detail in the manual page for *dbdbas(1)* for those who want to know all. Here are three commands that can be used from the command line:

- @ autoadd mode
- i insert a field description.
- d delete a field description.

The auto add mode always adds to the end. The insert and delete commands will prompt you for the information they need.

More on Database Creation

When you decide to create your own databases, you will need to do all of the things done here, and maybe more, if you want to use the other powerful features of *FirstBase*.

These extra features include things like on screen comments, help files, and auto indexes, and can be read about in more detail on the manual pages for *dbdbas(1)* and *dbedit(1)*.

Words of Encouragement



tips

Most people think that when they purchase a database management system, everything they ever wanted a computer to do will magically fall into place. This is not quite true — the magic is there, it just takes a bit of thought to make the magic work for you.

FirstBase provides a very powerful information processing environment. But you need to point it in the right direction by describing some of the pieces of information you want to manipulate.

But not to worry! You do not need to think of everything all at once! Decide on a few basics and work with those.

As you start using the *FirstBase* tools more, you may see that you need some additions to your earlier database definitions. *FirstBase* will comfortably evolve through these changes with you.

FirstBase is designed so that you can mold any of the data objects you build, adding or subtracting pieces as you decide they are needed or not. And as these changes take place, you will not lose any of the data you have already stored in *FirstBase*.

Again, the idea is to get at least part of what you want defined. And then play with *FirstBase*. You cannot build a mansion in a few minutes time. Work on the master bedroom first. Build in stages.

Database Considerations

Here are a few considerations you might want to look at. Some of the questions may be too complicated or inapplicable at this point. You can always come back to them.

When considering a database definition, the major goals and uses behind the need for the database should be thought about.

- Will this database be used for mailing labels as well as client billing and invoicing?
- Do you want the street address split into three fields — street number, street direction and street name — so that you can sort/select records by each of these pieces of information?
- Do you need to split the data into categories or ranking of some kind?
- Does each individual record in your database need to be uniquely identified by an account number field?
- Can you calculate some fields using numeric values stored in other fields?
- And how large should these fields be? Is 30 characters enough for the Last Name?

ideas

- Need space for an extension number after the phone number?
- Do you want to use the five or nine digit zip code?
- And what kind of information will be stored into each field? Will it be just numeric entries for the zip code? Or are foreign zips with alpha characters going to be included also?

If you define a field to be larger than 300 characters, you will need to know some form of UNIX editor to actually edit the field.

An extra hint is to add extra fields during the creation of the database for each changes later. This is not required since *FirstBase* can add new fields later. However, it is quite handy to have them quick access.

A little bit of thought here will go a long way, so do not be afraid to try something.

hints

Field Typing

Deciding what kind of data a field will be used to hold can be difficult. It is hard to know in advance all of the types of data that you will want to put there.

If a field is defined as numeric, you will never be able to put alpha characters there.

For instance, if the **Telephone** field of the Phone database was defined as type **N** for numeric, then we would not be able to put in “ext 45” since the characters **ext** are not numbers.

There are quite a few different types of fields that can be defined in *FirstBase*. These types include dollar (\$) and date (d) as well as the alphanumeric and numeric fields already discussed.

For a complete list of all possible field types, see the manual pages on *dbdbas(1)* and *dbedit(1)*. Furthermore, "Editing The Field" on page 5-15 describes the required input to each of these fields.

Alphanumeric vs. Numeric

One point worth noting is the inherent difference between alphanumeric fields and numeric fields.

Alphanumeric fields will always display as left justified values, whereas numerics are right justified. This type and display difference will cause slightly different scanning behavior during the use of the database editors.

Lets say you are designing a database where every record has a unique account number and you make the account number numeric.

When a numeric field is used as a search field, i.e. it is indexed and used to locate records, you will have to enter the value exactly as it appears in the record.

For example, to find account 1234, you must enter '1234' exactly.

In contrast, for the `LastName` field of our Phone database example, we could enter just the beginning of a last name to locate a record.

As an example, if you entered 'Smi', the beginning of **Smith**, then *FirstBase* would find the first record that begins with "Smi".

In the previous example, if the account number is alphanumeric instead of numeric, then you could enter '12' to find the '1234' record.

These choices depend on the application — what is the data and the database going to be used for. Again, the idea is to think a few minutes before blindly defining a database.

The Field Default

The Field Default is a value that will be inserted during data entry if a <RETURN> is entered without any data.

For example, if you have a field for `State` in a database containing mailing addresses, you could set the default to be **AZ** so that a <RETURN> will insert the characters "AZ" for you into the state field.

However, if you do not define a default value for a field using the database dictionary editor, then when you input that field, you will not have any options — you will have to enter a value there.

When a field does not have a default, it is called a *forced entry*.

If a default is defined, it will be used when you are creating new records and when updating old ones.

To make entry into a truly optional field, use a single blank as the **Default**. When this is done, the *FirstBase* editors will allow this field to be empty when creating or updating a record.

Blanks in the default string will appear on the screen as underscores when redisplayed by the database dictionary editor. A single blank will appear as a single underscore.

Refer to the detailed manual page on `dbdbas(1)` for a list of some variations on the default value which are quite powerful.

Summary

This chapter has discussed the more simple aspects of defining a database using the database dictionary editor. Still more details are described in the manual pages on *dbdbas(1)* and *dbedit(1)*.

Once you are comfortable with this dictionary editor, you will be familiar with all the other dictionary editors in *FirstBase* — they all behave in the same manner.

Remember that nothing in *FirstBase* is cast in concrete. This is the power behind *FirstBase* — you can continually construct and restructure your definitions and data.

Anything you define and enter can be converted to different sizes, types and locations. Let your database evolve as your knowledge of *FirstBase* grows.

Remember Thee!
Ay, thou poor ghost, while memory holds a seat
In this distracted globe. Remember thee!
Yea, from the table of my memory
I'll wipe away all trivial fond records,
All saws of books, all forms, all pressures past,
That youth and observations copied there.

Hamlet
WILLIAM SHAKESPEARE

And now, go, write it before them on a tablet, and inscribe it in a book.

Isaiah 30:8
BIBLE

The philosophers have only interpreted the world in various ways; the point is to change it.

KARL MARX

A successful tool is one that was used to do something undreamed of by its author.

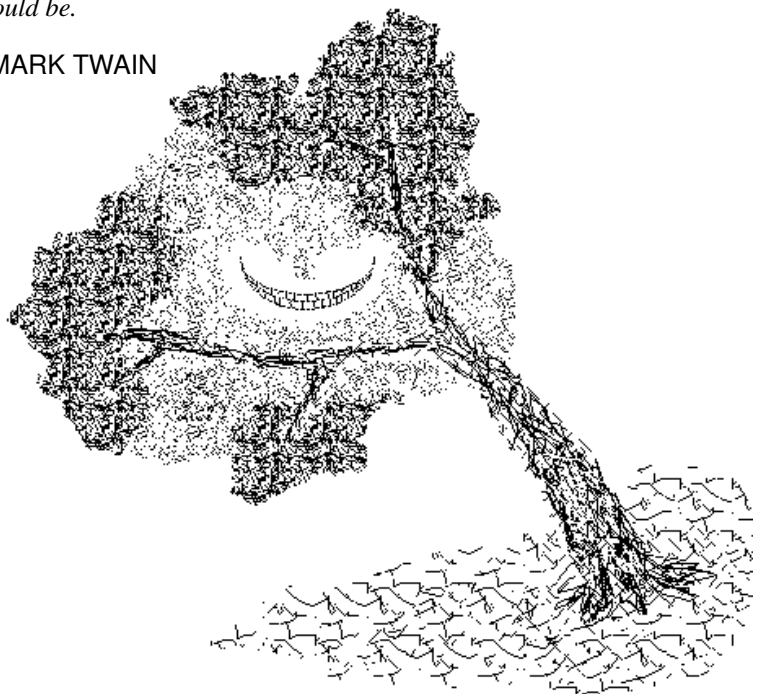
S. C. Johnson

When it is not necessary to change, it is necessary not to change.

LUCIUS CARY, VISCOUNT FALKLAND

Substitute "damn" every time you're inclined to write "very"; your editor will delete it and the writing will be just as it should be.

MARK TWAIN



The Database Editor

The heart of *FirstBase* is the database editor. Formally called *dbedit* (dee-bee edit), the database editor is used for creating, locating, and modifying records of a database.

This chapter describes the basics needed to begin using *dbedit*. A walk through example is provided for entering a few records also. For more extensive details, see the manual page for *dbedit*(1).

Concepts Behind *Dbedit*

Dbedit can be viewed as a specialized editor that allows database maintenance tasks on any *FirstBase* database. This tool is used to enter and modify all data stored in a database.

Dbedit will behave exactly the same on every database regardless of the database size and shape.

This consistent behavior is the idea behind a tool. *Dbedit* treats all databases the same way. The size and shape of the databases change, but the action and behavior of *dbedit* remains the same. *Dbedit* is a database editing tool.

The Database Revisited

A database is an organized package of information. It consists of many smaller packages of data called “records”. A record keeps logically connected pieces or fields together for quick access.

Formally, a database consists of one or more records, where each record consists of one or more fields.

Here is the sample Phone database from Chapter 4. It will be referenced many times in this chapter, and the following ones:

	Field 1	Field 2	Field 3	Field 4
	Last Name	First Name	Initial	Phone #
Record 1	Smith	Fred	B	299-8771
Record 2	Jackson	Seymour	P	327-8211
Record 3	Mills	John	J	749-2864

Overview of *Dbedit*

basic
concepts

Dbedit is used to create, locate and modify records in a database.

As a record is created, *dbedit* walks you through each of the fields of that record and allows you to enter data.

Each field is well labeled, so you will know what kind of item *dbedit* is expecting you to enter.

When you are done creating or locating a record *dbedit* allows you to page through the fields of the record, modifying any fields desired.

After you are done, the record can be deposited back into the database, and another one can be created or selected.

Dbedit is a straight forward, one field per line, database editor. It uses the full screen, but there are only ten fields per page, though you can have any number of pages.

There is a much more flexible version of *dbedit*(1) named *dbvedit*(1). The behavior of these two database editors is very similar in most cases. However, *dbvedit*(1) allows complete control over the display screen. This flexible screen is controlled by a view dictionary, described in *view*(5).

It is suggested that you use *dbedit*(1) first, then move on to *dbvedit*(1) once you are comfortable — and only if you need the extra capabilities of *dbvedit*(1).

So, even though there are two *FirstBase* database editors, this chapter approaches these editors as if there is only one.

Getting Started

name
database
and index

A database editor walk through example is fully described in "Walk Through Data Entry" on page 5-18.

For the moment, only a general series of steps leading to database editing will be covered.

First things first — you need to have your database (and maybe index) named properly. See the section "Changing a File Name" on page 3-5.

After naming your database, you need to start the database editor. However, the exact method for this task depends on the installation.

start
database
editor

If you have been presented with a series of menu selections, then you could be using a form of the *FirstBase* menu control system.

For example, if you are using the standard menu shell (MAIN), you would start using the database editor by selecting the **Edit a Database** command, **edit**.

edit

However, it could be that you have special selections that automatically use certain databases — in other words, custom menus. If so, you will have to know which selection invokes the database editor.

Or, you could have been given a command that will start the database editor for you, with all names properly defined.

Somehow, some way, you need to start the database editor using a properly defined database. *Dbedit* will loudly refuse to do anything otherwise.

If you are unsure of how to accomplish this task, see your computer system manager.

The Beginning: The *Dbedit* Command Screen

When you first start *dbedit*, your screen will be painted with the standard database editor command screen. This command screen is the only doorway into *dbedit* — any time you enter or exit *dbedit*, you will pass through a similar command screen.

Here is our simple Phone database *dbedit* command screen:

```
FirstBase 6.5: dbedit      FirstBase Software      Status: Command Level

1> Last Name [.]..... <a 35>
2> First Name <a 20>
3> Initial <a 1>
4> Telephone <a 15>

Files: Phone(0/0) index(0/0) [john: /usr/tutorial 15:25 01/16/92]
<CTL>-H=Help, -=End (dot@0/0)
```

add
modify
exit

Now What?

From the command screen you will do one of three things:

- Add a Record
- Locate/Modify a Record
- Exit *dbedit*

That's it. Deleting a record is considered a modification.

Each of these items will be well covered in the following sections.

A Sample Command Screen

Although the labels and other field information change from database to database, the basic *dbedit* command screen looks the same.

The screen on page 5-3 shows the command screen for *dbedit* when used on the Phone database depicted in the table at the beginning of this chapter.

The header line (discussed in "The Screen Header" on page 2-7) contains the status area that will always display the current editing level of *dbedit*.

There are only three levels, with Command Level being the highest, or outermost, level.

The last line on the screen is used to display a small list of commands that are available at this level, as discussed in "The Command Line" on page 2-8.

Each of these commands, and others, are detailed in "Dbedit: Command Level Mode" on page 5-5.

The Field Lines

In between the screen header and footer on the *dbedit* command screen, you will see one line for each field of the defined database.

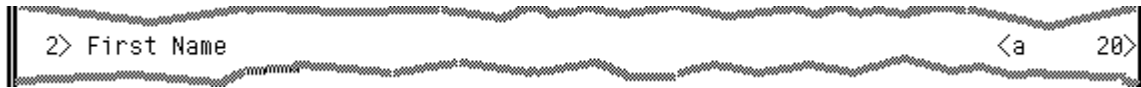
Only a group of ten fields may appear on a single "page" of the record. In other words, it takes ten fields to fill the screen with data.

This database has four fields, so the whole record template can fit on a single "page". However, there is no limit to the number of fields, or pages, a single record can contain.

Each field line is numbered and labeled on the left edge of the screen, with type and length information appearing on the right edge of the screen.

This leaves an empty area in the middle of the screen where the data will be input and displayed.

For example, the second field in the sample database command screen is shown as:



meaning:

- the numeral '2' can be used as a command to edit this field
- the field name or label is "First Name"
- the field is an alpha type field (a)
- the field can store a maximum of 20 characters

The Cursor

The cursor in the sample *dbedit* command screen on page 5-3 has been placed on the first field of the database.

Actually though, the cursor is placed on whatever field the index of that database refers to, which in many cases is the first field, but not always. So, do not get confused if the screen starts off with a different number than one.

Dbedit: Overview of Control Levels



The database editor has three levels or layers of screen displays. Each of these three levels accomplishes a different task.

The outermost layer of control is the **Command Level**.

The next level is the **Record Level**, with the innermost layer being the **Field Level**.

The next few sections discuss these three levels of control in detail.

Dbedit: Command Level Mode



The first layer of screen display for the database editor is known as Command Level.

Once this command screen is painted onto the monitor, *dbedit* is expecting some kind of action on your part.

Whatever you enter here is considered to be a *dbedit* command. Although there are many commands available, the basic three you must know to use *dbedit* are (again):

- Add a Record
- Locate/Modify a Record
- Exit *dbedit*

command
level

add

Adding New Records - '@'

To add records to a database you must go into “Add Mode”. This is a screen level “below” the command screen.

You will stay in Add Mode until you are done adding records. When done, the END key ('-', the dash key) is used to exit Add Mode.

@

To get into add mode, the command sequence '@' is used from the *dbedit* command level screen. This command tells *dbedit* that you have one or more new records to create and enter.

Once into Add Mode, the Phone database screen will look like:

```

FirstBase 6.5: dbedit          FirstBase Software          Status: Field Level
                                Add Mode
1> Last Name [.].....          <a      35>
2> First Name                   <a      20>
3> Initial                      <a       1>
4> Telephone                    <a      15>

Files: Phone(0/0)  index(0/0)  [john: /usr/tutorial 15:27 01/16/92]
Enter Data, -=END, <CTL>-D=Defaults, <CTL>-H=help          (display@1/4)

```

The command line displays the only choices you have at the moment.

Note that one of these choices is the END keystroke. This keystroke will only work on the first field of the Add Mode screen — and then, only to return to the Command Level.

basic
concepts

After you get past the first field, the above command help line will change slightly, and list all but the END key as choices.

The idea is to first enter all of the fields of one record, then to correct any mistakes that were made. To automate this task, *dbedit* takes you through each field, and requires a value to be entered into the blank field line.

During this data entry phase, the kind of input required varies depending on what type of field you are entering. The different input types and how to deal with them are covered in "Editing The Field" on page 5-15.

Once all the fields have a value in them, *dbedit* goes into what is called the **Record Level Mode**, which allows changes to any of the fields in the current record.

This mode is where mistakes from the initial entry can be corrected. See "Dbedit: Record Level Mode" on page 5-11 for complete details.

Per usual, to exit the Record Level Mode, the standard END key '-' is used. This physically stores the record into the database, and resets the screen to add more records.

Again, to exit this Add Mode, you must use the END key '-' while the cursor is on the first column of the first field. This keystroke will return you to the *dbedit* command screen, ending your session of adding records.

exiting
add mode

After you read through this chapter, you might want to try and add the information from the simple Phone database into an actual *FirstBase* database.

The section "Walk Through Data Entry" on page 5-18 describes this task in detail.

When done adding records, use the END keystroke to get back to the Command Level.

Locating/Modifying A Record - <key>

To locate a record that has been previously entered into the database, you must start at the screen as shown in "The Beginning: The Dbedit Command Screen" on page 5-3.

It is assumed that your database is autoindexed (or at least indexed) with the field you want to use to search for a record.

To start the search for a record, you must enter some characters, called a *key*, to help *dbedit* find the record. Depending on the type of the field, you may have to enter in the exact key to find a match.

In our simple phone database example, the Last Name is an alphanumeric field. This means you would not have to enter an exact match of the record you are looking for.

command
level
modify

key

For example, if you enter 'Jac' at the *dbedit* command screen, *dbedit* will look into the index and database, and quickly determine if there is a record that has a Last Name that begins with **Jac**.

If all of the records from the sample Phone database have been input, then *dbedit* would respond with a screen like:

```

FirstBase 6.5: dbedit          FirstBase Software          Status: Record Level
                                *00001:Jackson
1> Last Name  Jackson                <a    35>
2> First Name Seymour                <a    20>
3> Initial   P                      <a     1>
4> Telephone 327-8221                <a    15>

Files: Phone(3/0) index(0/3) [john: /usr/tutorial 15:33 01/16/92]
Field # (<CTL>-H=Help, -=End): . . . (display@1/4)

```

basic
concepts

Once a record is located that matches your search <key>, *dbedit* places the record into Record Level Mode. This mode allows changes to any of the fields in the current record. See the sections "Dbedit: Record Level Mode" on page 5-11 and "Dbedit: Field Level Mode" on page 5-13 more details.

exiting
a
located
record

Per usual, to exit the Record Level Mode, the END key '-' is used. This physically stores any modifications made to this record into the database.

After you exit Record Level Mode, you will be back where you started: the *dbedit* command screen. As mentioned, here you can locate another record, add more records, or exit.

A <RETURN> is considered to be a special search *key* that requests the next record. This is a method of single stepping through the records. If an index is used, the records are presented in index order. Otherwise, they are in database order.

command level

exit

—

other command level commands

<CTL>-X

/

<RETURN>

<CTL>-P

%

\$

Exiting the database editor - '-'

The last of the basic three *dbedit* Command Level commands has been mentioned all along in this chapter. It is the END key, '-'.

From the Command Level, the END key exits the editor session.

From the Record Level, the END key returns to the Command Level.

From the Field Level, The END key returns to Record Level.

More Simple Commands

All the other commands available at Command Level are very simple. To get a list of available commands, you can type <CTL>-H (Control H). This will give you a help screen with lots of detail.

Key searches for records in *dbedit* always start at the point you last stopped searching. Sometimes this behavior is not desirable.

The command level sequence <CTL>-X will put you at the “top of the index”. This means the next search request will begin at the top.

Dbedit does not always start at the top of an index for a search so that multiple “Smith”s, for example, can exist in the same index and still be found.

The second search for “Smith” will find the second occurrence of “Smith” in the database index. A shortcut for this search request is the search pattern '/' (backslash).

Dbedit will wrap at the end of the index file when searching for a record. This means that if the bottom of the index is hit before a record is located, the search will automatically continue from the top.

Also, the <RETURN> key can be used as an empty <key> to tell *dbedit* to locate the very next indexed record. Thus, you can find “Smith” once, and <RETURN> through all the others.

Other simple commands from command level include <CTL>-N and <CTL>-P, used to locate the next and previous indexed record.

Still other commands are record location commands. The '%' key as a command will locate the first indexed record. Alternately, the '\$' key can be used as a command to locate the last indexed record.

Remember after each record is found, you are put into Record Level Mode, and cannot get out until you type the END key '-'.

More Record Location Commands

(These commands are not needed to use *dbedit*, but might be useful after you are more comfortable with the database editor. If this is your first reading of this chapter, skip ahead to "Dbedit: Record Level Mode" on page 5-11.)

Some other useful record location commands involve regular expression pattern matching, case insensitive searching, and searches on non-indexed fields.

Regular Expression Record Searching

regular
expression
searching

FirstBase provides many methods for locating records using regular expression pattern matching on any field stored in the database. Only one of these methods involves the database editor directly.

To do interactive regular expression searches using the database editor, the *FirstBase* environment variable REGEXP must be set to ON.

This environment variable can be defined in one of three ways: the setup file, *.firstbase-init*, as specified by *setup(5)*; the UNIX command line, again described in *setup(5)*, or interactively.

To set REGEXP interactively, use the *dbedit extended command* keystroke, <CTL>-X <CTL>-X (or two ABORT signals).

These keystrokes will provide a miniature command line buffer at the bottom of the screen. The command to type is **set REGEXP ON**, followed by a <RETURN>.

Once REGEXP is ON, searches for records from the *dbedit* Command Level are done *sequentially* using the UNIX regular expression pattern matching mechanism, and are slower than *indexed* searches.

Note that these searches occur on the indexed field when *dbedit* is invoked with an index, but can be used on any field if invoked without an index. See "Searching on Non-Indexed Fields" on page 5-11.

case
insensitive
searching

A special case of regular expression searching, case insensitive searching, can be utilized by invoking the **-C** option on the UNIX command line when invoking *dbedit*. See *dbedit(1)* for details.

Regular expressions can also be used in index creation. Indexes are covered in Chapter 8, "Creating an Index", and in the standard manual pages *dbdind(1)* and *dbigen(1)*. The index generator also provides a case insensitive switch.

Additionally, other *FirstBase* tools that provide regular expression pattern matching features are *dbawk(1)*, *dbmacro(1)*, and *dbsql(1)*.

Each of these tools can be used to locate records.

Searching on Non-Indexed Fields

non-indexed
searching

The *FirstBase* database editor provides a method for searching on non-indexed fields as well. Non-indexed searches are *sequential*.

The first step is to invoke *dedit* without an index. To do this, use an index name that does not exist, or do not use the **-i** argument from the UNIX command line (and make sure there is no object named *index* in the current working directory, i.e. no *index.idx* file).

If you want to suppress the no-index warning from the database editor, use the UNIX command line switch **-w**.

>

Once the database editor is invoked without an index, the **>** and **<** keystrokes can be used to move the sequential search field focal point to the next or previous field on the screen.

<

Sequential searches are done using the contents of the current field of each record, and can take a long time. Sequential searches are and are interruptible.

Use this feature with REGEXP ON for best results. See "Regular Expression Record Searching" on page 5-10.

Dbedit: Record Level Mode

record
level

Once *dedit* has a complete record, either by adding a new one or by locating an old one, you will be put into what is called Record Level Mode. A picture of this screen is on page 5-8.

Viewing a Record: <RETURN>, 'b',

basic
concepts

In Record Level Mode, you are allowed to move around the record, paging through the fields in the record. There are a myriad of commands available here, but only a few are needed to get started.

A simple **<RETURN>** will display the next page of fields if it exists. If not, nothing will occur. The 'paging' wraps, meaning after the last page the first page is displayed again.

<RETURN>

The **b** command will take you one page backwards, if a previous page exists. Again, nothing will happen if there are not more than one page of fields. Backwards paging also wraps.

b

The last movement command is the **#**. It is meant as a reminder that the number to the left of the field name, the address of the field line, can be used as a movement command. However, be careful if you use this command to move from page to page.

#

The idea is that if the field you indicate with the number is **not** on the page being shown on the screen, then the page with your selected field will be displayed, and you will stay in Record Level Mode.

If however the field is on the screen, you will be put into Field Level Mode on the indicated field. This may or may not be what you intended.

Exiting Record Level: END, <CTL>-X

exiting
record level

Once you are done with a record, going into the Field Level Mode as many times as you desire, scanning and modifying, you can exit the Record Level Mode using the END key '-'. This will save all the changes you have made to the current record.

—

If you were placed in Record Level Mode automatically from Add Mode, then the record will be created, and the command line will show this.

But, if you are in Record Level Mode from your own selection, and you have made changes to the record, then the command line will briefly display "writing" instead of "creating" as you exit this level.

If you have accidentally made changes you do not want, or have added this record and now have changed your mind, or for any other reason you do **NOT** want to keep the changes made to this record, you can still recover.

<CTL>-X

Use the abort sequence, <CTL>-X, to escape Record Level mode without making any changes. *Dbedit* will noisily state that the record is not being updated before returning to where it was previously.

Deleting The Record: 'del'

del

Sometimes it is necessary to actually delete the record that you have located. To do this type the record level command 'del'. You will be asked to verify that this is what you want to do.

Note that the record is marked for deletion, but not physically removed from the database or any existing indexes.

See Chapter 11, "Removing and Restoring Deleted Records", and *db-pack(1)* for more information on how to pack these deletions.

Note, however, that new indexes generated after the deletion will not refer to the deleted record

Dbedit: Field Level Mode

This mode is where you will actually store information into a field of a record. The information may be new if the record is being created, or it may exist already, and you are just modifying it.

Entering And Exiting Field Level

**field
level**

To get into Field Level, you must either add a new record or find a previously created one.

If you are adding a new record, you will automatically be placed into Field Level mode (or more exactly, Auto Field Level) exactly once for each field, then into standard Record Level mode.

If you have located a record using a search key, then you will get into Field Level by one of two methods:

**basic
concepts**

- use the number associated with each field as a command, or
- use the next field (<CTL>-N) and previous field (<CTL>-P) commands. This actually invokes Auto Field Level, an augmented Field Level which allows movement *between fields* using the next field and previous field keystroke commands.

This Auto Field Level is also used for new records meaning you can back up during the *first* pass through the data entry.

Once in Field Level Mode, the cursor is positioned on the line for that field, and you are allowed to fill in the blank represented by the dots.

When *dbedit* goes into the Field Level mode on a field, the remainder of the screen is mostly untouched (unless it is an externally edited field, or a choice field, as seen in the field types section "Choice - c" on page 5-15).

Note: this normal database editor behavior is extensible and can be changed by using a macro field definition and the visual database editor, *dbvedit*(1). Again, a macro can be defined to do trigger actions and complex defaults. See *macro*(5) and "Macro Fields" on page 19-3

The following sample Field Level screen is from the Phone database:

```

FirstBase 6.5: dbedit          FirstBase Software          Status: Auto Field Level
                               *00001:Jackson
1> Last Name  Jackson                <a    35>
2> First Name [.].....                <a    20>
3> Initial   P                       <a     1>
4> Telephone 327-8221                 <a    15>

Files: Phone(3/0) index(0/3) [john: /usr/tutorial 15:33 01/16/92]
Enter Data, -=END, <CTL>-C=Abort Field, <CTL>-H=help          (display@2/4)

```

The numeral ‘2’ is the “address” of the field we want to modify.

The command line indicates that you must either enter some data for that field, use the abort sequence to get out without making changes, or use the help key to page a help file to the screen.

If you enter a value (data), you must use a <RETURN> to complete it. You will not be allowed to enter characters beyond the dots.

Note that the END key ‘-’ will exit from Field Level only if the cursor is located in the first column.

Also, if you are using the Auto Field Level, the next field and previous field commands (<CTL>-N and <CTL>-P) will work.

Otherwise, you must either complete the field or abort the editing of that one field by using <CTL>-X. Furthermore, <CTL>-H will display a user defined help file.

Once you are done entering data for a field, the next task depends on how you got into Field Level mode.

If you are adding a new record, *dbedit* will go straight into Field Level mode on the very next field.

Alternately, if you located this record from Command Level, *dbedit* will go back to the Record Level mode.

exiting
field level

details
on
types
of
fields

Editing The Field

Many factors control what kind of input and action will be allowed for each field during the actual entering of data into a field.

The type of the field determines what can be entered. A field can also have a default value, some value that will be stored in the field if you choose not to enter anything.

Fields can be long or short, point to other places for help, or consist only of a formula that depends on the values of other fields.

The action and reaction of every type of *dbedit* field is detailed below.

The single key letter listed with each type is the letter that will appear on the right edge of a field description line of that type.

Alphanumeric - a

a

Any kind of input is accepted, characters, numerals, punctuation, etc. This is the catch-all type for a field.

Fields of size 50 characters or less are edited in place. Fields between 50 and 300 characters are edited using an internal sub editing mode. This length of fields does not allow newlines in it.

Finally, there are fields greater than 300 characters. These fields, sometimes called *visual* fields, are edited using a UNIX text editor.

Alpha - A

A

Only characters that are upper or lower case alphabet characters are accepted. Formally this is A-Z and a-z. Nothing else is accepted.

Binary - b

b

Unlike all other fields within *FirstBase*, the binary data type is a fixed length field. These fields will be padded to the field length.

A database with a binary field should be used for read only storage since these databases *cannot* accurately be cleaned via *dbclean*(1).

Choice - c

c

Choice input takes the user to a new screen and displays a list of choices that are possible at this point.

Each choice has a meaning next to it.

The meaning of the choice is what will be input into the database — not the choice itself.

For example, suppose the choice screen looks something like:

Choice	Meaning	Comment
1	Red	The color is red.
2	Blue	The color is blue.
3	Green	The color is green.

If you choose <2>, followed by a <RETURN>, then the characters “Blue” will be entered for you into the proper field of the database.

The choice screen is then exited, and the previous *dbedit* screen is re-displayed. See *choice(5)* for details on building choicefiles.

Silent Choice - C

C

Same as type choice, but there is no screen display.

Date - d

d

All dates can be entered into *FirstBase* in MMDDYY format. Slashes and dashes can be entered if desired.

The date input, if a proper, valid month/day/year, is redisplayed in MM/DD/YY format.

Dollar - \$

\$

The dollar data type means that only numerals are expected. Also, the decimal point is assumed (although this can be disabled by using the DECIMAL variable described in *setup(5)*).

This means you enter “10000” for one hundred dollars. All else is an error.

Correct entries will be redisplayed with a decimal and embedded commas.

Negative dollar amounts will appear in parenthesis or with a leading negative sign (depending on NEGATIVE as defined by *setup(5)*).

Floating Point - f

f

Floating point means a number that can have fractions, such as “4.37”. For type “f”, numerals, the “.”, and a leading minus sign are accepted. Nothing else.

You must use two END keystrokes if you want the END key signal.

Formula - F

F

Formula types are strange — the fields they represent do not exist in that they do not take up any space.

These fields are used to calculate and formulate information from other fields into one answer.

For example, a field called “TotalPay” might be the NumberHours field multiplied by the PayRate field.

Again, these fields do not physically take up space. Thus there is no input into a formula field.

Link - L

L

Links provide a dynamic join feature that allows fields of one database to point to fields in another database. The fields will be display only. For more information, see *dbdbas(1)*.

Numeric - n

n

Only numerals are accepted. A leading minus sign is allowed.

You must use two END keystrokes if you want the END key signal. Anything else is an error.

Positive Numeric - N

N

Only positive numerals (numbers greater than or equal to zero) are accepted. All other entries result in error.

Upper Case Alphanumeric - U

U

This type is the same as type “a”, except that all lower case letters are automatically converted to UPPERCASE.

This feature works in fields of any size, and can be used to facilitate the searching of fields for word patterns.

Extended Choice/Help - X

X

Extended choice fields are choice fields that dynamically produce choice lists during editing. These choice lists are stored in a distinct *FirstBase* database, and can even be filtered. See *dbdbas(1)* and *xchoice(5)*.

Walk Through Data Entry

To follow along with this walk through example, there are a few things that you need to make sure have been done:

- The first assumption here is that you have completed the Phone database dictionary defined in Chapter 4 of the *FirstBase User's Guide* exactly as it is defined. Complete with the AutoIndex described.

preliminary
assumptions

If you have not completed this task, do so. This walk through example relies on the Phone database for its examples.

- The second assumption is that you are using the *FirstBase* menu tool and have defined the database name in your current *FirstBase* menu to be **Phone**.

If you are continuing from Chapter 4 of the *FirstBase User's Guide*, chances are you still have the correct database name, 'Phone', defined properly. If so, it will appear in your footer line, on the left side, next to the word 'Files'.

If you do not have the database named properly, see "FirstBase File Names" on page 3-5.

<CTL>-E
d
Phone

Here are the keystrokes if you do not want to look back at Chapter 3: From any of the *FirstBase* menus, enter <CTL>-E, 'd', 'Phone', <RETURN>, <RETURN>.

To start using the database editor, select the **Edit a Database** command, **edit** from the MAIN menu.

edit

Since this database has never existed before, we will actually be creating the database — just by using the database editor. You will get a message on the screen about creating a database this time. But not again.

If you failed to define the autoindex as described, or if you failed to name this the default name of **index**, you will get a warning that you are not using an index.

If this warning happens, enter a <RETURN> to get to the command screen, and the END key to exit back to the menu.

Then define the AutoIndex part of the Last Name field as described in "AutoIndex the Database" on page 4-8, and start the walk through demonstration again.

Adding A Record

To add a record, you will start at the *dbedit* Command Level screen as shown on page 5-3.

@

Here you will type the at sign, '@' and a <RETURN> to place the database editor into Add Mode.

The status area in the header line (far right side) will show that you are in Field Level as well as Add Mode.

Now enter the Last Name field — use 'Jackson', and a <RETURN>. As you hit the <RETURN>, the input you finished will be redisplayed — in reverse video if possible.

enter
data

The cursor will move on to the First Name field. Enter 'Seymour' here, and a <RETURN>.

Again, the field will be redisplayed in reverse video, and the cursor will move to the next input area, the Initial field.

Enter a 'P' into the Initial field, a <RETURN>, and then '327-8211' and a <RETURN> at the Telephone field.

At this point you will have the screen depicted on page page 5-8.

Corrections

making
corrections

Say you misspelled 'Seymour' and needed to re-enter it. Then you would merely type a '2' and a <RETURN> from the Record Level screen and correct the field information.

The screen will appear like that shown on page 5-14.

You can even correct a correction.

Once you input the '2' and <RETURN>, if you decide to keep the information that was previously there, you can type a <CTL>-X to abort out of the Field Level without making changes.

Saving Your Record

Once you have entered a complete record, you will be placed in Record Level.

—

As outlined in this chapter, from here you make any needed corrections.

Then use the END keystroke to exit the Record Level.

enter
more
data

This END keystroke will return you to the Add Mode for entry of more records.

Go ahead and enter the other two records that are shown in the table at the beginning of this chapter.

Exiting Add Mode

After you are done adding records, you may want to locate some of these records you have added, or other records that were there, or even exit the database editor.

To do these tasks, you need to go back to the Command Level.

To get to the Command Level screen from the Add Mode, Field Level screen, use the END keystroke.

This END keystroke will return immediately to the Command Level screen.

Summary

There are many more complexities to *dbedit(1)*. What has been listed here is an attempt to get you going with the database editor.

The *FirstBase* database editor is not hard to learn if you can read and follow along on the terminal at the same time.

There is a quick on line help screen that might be helpful once you understand the motions of the database editor.

For instance, you can look here for the commands to quickly locate the first or last record of a database.

Help is available from command and record level modes within *dbedit*. To get this simple help screen, type the help sequence <CTL>-H.

For all of the details to the more advanced aspects of *dbedit*, see the manual page for *dbedit(1)*.

'What is the use of a book', thought Alice, 'without pictures or conversations?'

Alice's Adventure in Wonderland
LEWIS CARROL

The poet ranks far below the painter in the representation of visible things, and far below the musician in that of invisible things.

LEONARDO DA VINCI

*Write the vision; make it plain upon tablets,
so he may run who reads it.*

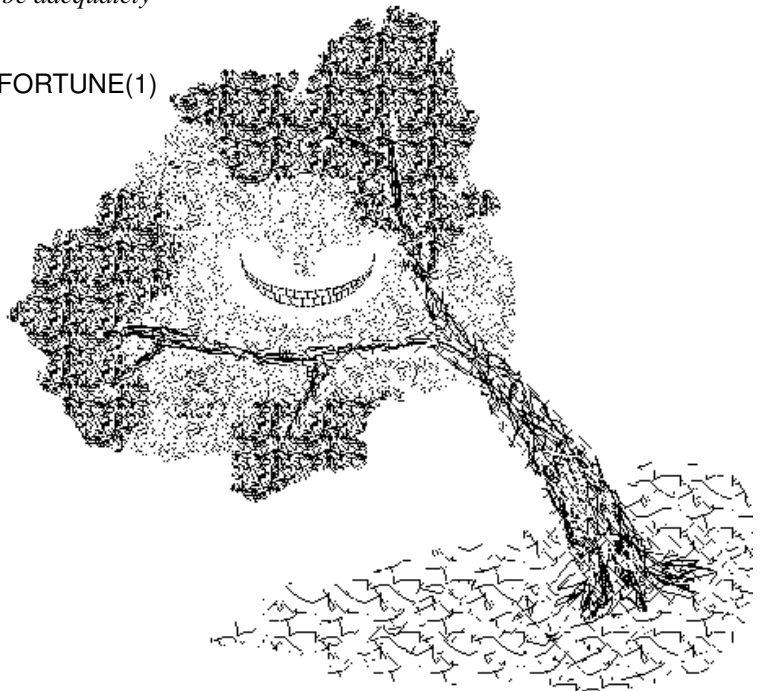
Habakkuk 2:2
BIBLE

*'Tis distance lends enchantment to the view,
And robes the mountains in its azure hue.*

THOMAS CAMPBELL

Re: graphics: A picture is worth 10K words — but only those to describe the picture. Hardly any sets of 10K words can be adequately described with pictures.

FORTUNE(1)



Creating Custom Screens and Views

This chapter details how to define a custom screens and views for a *FirstBase* database.

A custom screen for *FirstBase* merely rearranges selected fields of a database for the database editor, *dbedit(1)*. The tool used to define screens is *dbdscr(1)*. This tool is detailed in part of this chapter.

The second part of this chapter provides some direction for those wanting to build the more complex view dictionaries for use with the *FirstBase* visual editor, *dbvedit(1)*.

Database Screens

basic
concepts

FirstBase has a mechanism that allows each user to select which fields of a database they want to see when using the database editor, *dbedit(1)*. The order of the fields can also be specified.

Screens are defined and stored in screen dictionaries. There is no real generator that goes with the Define Screen program, although the database editor could be seen as the generator.

These screen dictionaries are the mechanism that provides a physical to logical mapping of the database fields. Not all fields need to be used in the screen dictionary — making it easy to hide fields from those who do not want to see them.

Screen dictionaries are used mainly with *dbedit(1)*. If a screen is not defined, the database editor will display all defined fields.

These screen dictionaries have no affect on most other *FirstBase* tools. However, there are exceptions.

Screen dictionaries can be used with *dbemit(1)*, *dbdmrg(1)* and *dbawk(1)* to provide a set of ordered fields to work with in each of these tools.

Furthermore, screen dictionaries can be used with *dbvedit(1)* for custom commands. See these manual pages for more details.

This screen mechanism means you can define a database to have extra fields and then screen these fields out for the moment. If you ever need a new field, you can put that field in easily, without running the Database Conversion programs, since it is actually already there.

The Screen Dictionary

The screen dictionary is merely a list of database field names. The field names are cross referenced in the database dictionary to make sure that they exist.

The idea is to build a list of database field names in the order that you want them to appear during database editing with *dbedit*(1).

UNIX Note

For all the UNIX oriented people: a screen dictionary is a list of field names, one per line, no extra lines, with underscores substituted for any blanks in the field name.

Use of a text editor or other UNIX tools on any of the standard *FirstBase* dictionaries is permissible and is a feature, but should not be done by the casual user — in any case.

The relevant manual page is *dictionaries*(5).

The material presented here discusses the *FirstBase* full screen database screen dictionary editor, *dbdscr*(1), which checks on the existence of fields and does the underscore substitution.

Getting Started

<CTL>-E d Phone

From the menus, you will need to make sure the database is named properly, and that a screen has been named using the same method. See "Changing a File Name" on page 3-5.

(Here are the keystrokes if you do not want to look back at Chapter 3 of the *FirstBase User's Guide*: From any of the *FirstBase* menus, enter <CTL>-E, 'd', 'Phone', <RETURN>, <RETURN>. Easy. This will name the database. You can Figure out how to name the screen.)

Now, locate the 'Define Database Screen' selection in the *FirstBase* MAIN menu. The command you enter is **screen**.

The 'Define Screen' Screen

The following screen fragment shows what the top of your screen will look like when you start the Define Screen tool:

```

FirstBase 6.5: dbdscr          FirstBase Software          Status: Define Screen
Field          Default
-----

```

The Field Column

To begin entering update fields, use an '@' to go into 'auto add' mode. In this mode, you will continually enter fields until the standard END keystroke is used from the field column.



The cursor will be placed under the **Field** column and you will have to enter a valid database field name. Then hit <RETURN>.

Enter only the fields that you want to see during database editing — in the order you want to see them.

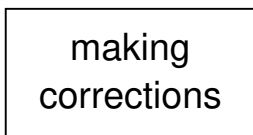
The Default Column

Next, the cursor will be placed under the **Default** column. Here you can enter a default that is different than the one stored in the database dictionary. This means that different screens can have different sets of field default values.

Note: there is a more general method of storing sets of defaults for the same database. Defaults stored in this more general method will be seen by all tools, not just those using a screen dictionary. For more information, see *defaults(5)*.

Making Corrections

To exit add mode, use the standard END keystroke '-' from the field column.



This keystroke will put the cursor back onto the any change prompt at the bottom of the screen

Like all other *FirstBase* tools, you can make corrections to any of the dictionary lines before you exit from the tool.

A correction is done by typing the number of the line you want to change at the Any Change prompt.



Use the END keystroke to exit the Define Screen program.

The following fragment represents a completed screen dictionary using the sample Phone database fields:

```

FirstBase 6.5: dbdscr      FirstBase Software      Status: Define Screen
  Field      Default
  =====
  1) First Name
  2) Last Name

```

If this screen dictionary is used with the database dictionary as defined in chapters 4 and 5 of this manual, the result would mask out the Initial and Telephone fields, as well as reverse the order of the First Name and Last Name fields.

Exiting the Define Screen Tool

When you exit you will be asked whether you want to run the database editor. If you respond with a **y**, the Database Editor will be run using the screen dictionary you have just defined. (Provided you named your screen properly from the *FirstBase* menus).

Otherwise, you can always reset the screen name and run the database editor from the MAIN menu. Again, this is just for *dbedit(1)*.

Database Views

Database views require a much more complex dictionary that maps both fields and text to the monitor or output file.

View dictionaries are used by the *FirstBase* visual editor, *dbvedit(1)*, as well as some other tools like *dbvemit(1)*, *dbvi(1)*, and *dbvform(1)*.

Additionally, only *dbvedit(1)* and *dbvemit(1)* support extensible editor behavior using macro field definitions (see *macro(5)*), so a view dictionary will be required for applications that want macro features.

A view dictionary is a text file that is edited using a UNIX editor. There are many features to a view dictionary file. These are outlined in the manual page *view(5)*.

getting
started

makeview

There is a quick way to generate a basic view dictionary. This is done by using the *makeview*(8) command.

Makeview reads a given database dictionary and converts it into a simple view dictionary that can be used as is or as a basis to start designing your own custom view dictionaries.

Read more about this command in *makeview*(8).

Additionally, there are some helpful hints for more advanced topics within view dictionary building located in "View Dictionary Hints" on page 19-3.

Summary

This chapter has discussed the process of defining a screen dictionary for use with the database editor. This allows users to reorder fields for their own convenience. See *dbdscr*(1) for more details.

Also discussed here were view dictionaries and their use by the *FirstBase* visual editor, *dbvedit*(1). See *view*(5) for more information.

And the more words
All were the
The more words
The more words

Lewis Carroll
Through the Looking Glass
Lewis Carroll

*What is a modern poet's fate?
To write his thoughts upon a slate;
The critic spits on what is done,
Gives it a wipe — and all is gone.*

THOMAS HOOD

*A computer, to print out a fact,
Will divide, multiply, and subtract.
But this output can be
No more than debris,
If the input was short of exact.*

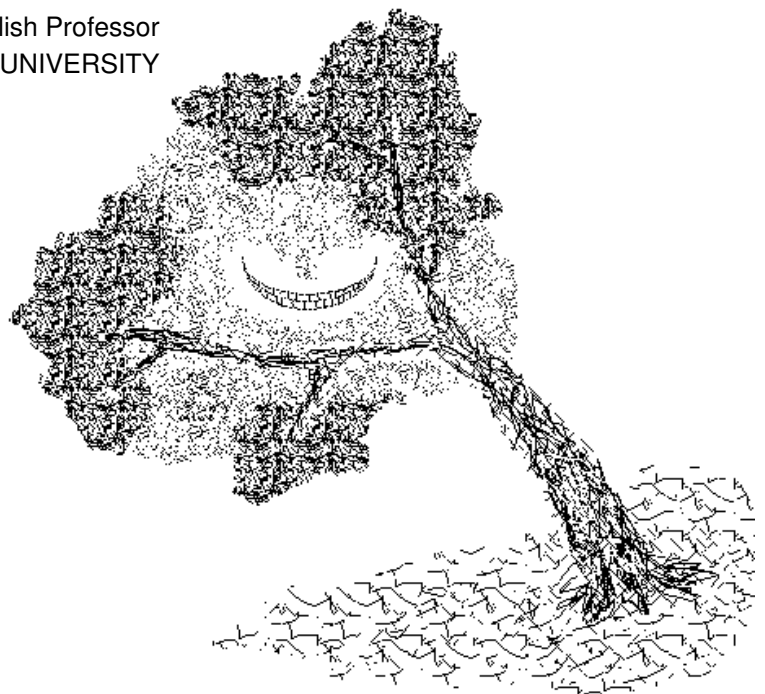
Gigo
FORTUNE(1)

Be careful of reading health books, you might die of a misprint.

MARK TWAIN

*"I am returning this otherwise good typing paper to you
because someone as printed gibberish all over it and put
your name at the top."*

English Professor
OHIO UNIVERSITY



Creating *FirstBase* Reports

The first part of this chapter describes one quick method of generating an organized report, complete with title, sub headings and page numbers. The *FirstBase* tools covered here are *dbdprt(1)* and *dbpgen(1)*.

The second part of this chapter provides some direction for those wanting to build more complex printouts using some of the other tools in *FirstBase*, including *dbvform(1)*, *dbsql(1)*, *dbawk(1)*, *dbmacro(1)* and *dbemit(1)*.

Defining a Report

A *FirstBase* report is a well formatted, tabular listing of selected fields from a database. You can control some formatting items like line spacing and page width.

FirstBase will take care of indenting, column headers, page formatting, and most other aspects of the report layout.

In most *FirstBase* documentation, the word **report** and the word **printout** are used interchangeably.

Getting Started

basic
concepts

To create a report, you need to define or list the fields from the database that you want the printout generator to use. This task will be very similar to the creation of a database dictionary — except you will be defining a printout dictionary.

If this is your first time through this chapter and you are planning on following the examples at your terminal, there are a few things that you need to make sure have been done.

preliminary
assumptions

The first assumption here is that you have completed the Phone database dictionary defined in Chapter 4 exactly as it is defined. Complete with the AutoIndex described. If not, do so. This chapter relies heavily on the Phone database for its examples.

The second assumption is that you have defined the database name to be **Phone**. If so, it will appear in your footer line. If not, see "Changing a File Name" on page 3-5.

The third assumption is that you have entered all three records as shown in "The Database Revisited" on page 5-1.

2

To begin defining the report, choose the **Define Printout** selection from the *FirstBase* MAIN menu, selection <2>. This will start the printout dictionary editor.

Creating the printout definition

The following screen fragment shows what the top of your screen will look like when you start the Define Printout tool:

Field	Field Width	Line Length	Total?	Break?	Level
=====	=====	=====	=====	=====	=====

The bottom of the screen will have a standard Any Change prompt.

From here you will choose fields that you want to appear in the printout. These fields will appear in the same order as you define them on the screen.

@

To begin this entry, you will go into auto add mode. (Just like the database dictionary editor — just like the database editor itself.)

Use the at sign '@' to begin auto add mode.

The Field Column

The first screen you see after you begin add mode will probably be the standard Choose Field screen. This common screen is detailed in "FirstBase Choose Field Mechanism" on page 2-12. You can either select the fields you want, or use the method described below.

easiest
method

The Easiest Method

Everyone wants to know the easy way out, right? Here it is. You are in auto add mode for the printout dictionary editor, and the *FirstBase* tool is waiting for your selection or entry of a valid field name, right?

Now hit the <RETURN>. This action will enter the first field for you, and move the cursor to the 'break' column.

8 <RETURN>s

Now hit the <RETURN> again, and the cursor will be back at the second row or choose field screen, waiting for more entry.

So, hit the <RETURN> two more times for this line. And two more for the next line. And two for the fourth line also, for a grand total of eight (8) <RETURN>s.

—

Then hit the END key '-'.

At the end of these 8 <RETURN>s (using the Phone database exam-

ple), you would have a screen that looks like the following fragment:

```

FirstBase 6.5: dbdprt          FirstBase Software          Status: Define Printout
  Field      Field Width      Line Length      Total?      Break?      Level
  ----      -
1) Last Name      35              37 (1)
2) First Name     20              58 (1)
3) Initial        7               66 (1)
4) Telephone      15              16 (2)
    
```

Field Totals and Breaks

Numeric, floating and dollar fields can be totaled for any printout. A special subtotal can be forced to print when a field changes values. The field whose value causes the subtotal to be printed is called a break field.

Multiple level breaks for any field may be defined. A break on any level causes subtotals for that break level and all higher break levels to be printed and reset to zero (0) when that field changes values.

Corrections and Features

Like all the 'editors' in *FirstBase*, you will get the Any Change prompt and be allowed to change data and correct errors that appear on the screen.

making corrections

If you followed the above example, entering 8 <RETURN> key-strokes and the END key, you probably do not have any errors.

But, like other *FirstBase* tools, you would enter the number next to the item that you want to modify. The cursor will go from column to column, allowing new changes to be inserted. To skip a column, thus keeping the value it had, use the abort keystroke, <CTL>-X.

You can use the dictionary editor commands **d** and **i** to delete or insert dictionary items.

d

3

To make these example work well, delete Item # 3, the Initial field.

Enter a '**d**' at the Any Change prompt on the command line, and then respond to the question "Delete what number?" with the numeral '**3**', and a <RETURN>.

The screen will be redisplayed without the Initial field item. The numbers under the Line Length column will change slightly.

Miscellaneous Report Information

The miscellaneous screen contains variable information that you can control for each individual report. You will be taken to this screen automatically as you exit the printout dictionary editor.

m

Alternately, you can use the **m** command to get there.

At first, the miscellaneous screen looks something like this:

```

FirstBase 6.5: dbdprt          FirstBase Software          Status: Define Printout

Miscellaneous Information

Title:          Sample Phone Printout
Output File:    cdbprt
Spacing:        1
Columns/Page:  80
Print Detail?   Yes

Files: Phone(3/0)  index(0/3)  [john: /usr/tutorial 18:19 01/19/92]
Any Information Change (y/<RET>) .

```

Enter **y** at the Any Change prompt. This will move the cursor over each of the variable data areas allowing you to change their content.

Again, if you use **<CTL>-X**, the value is not changed.

Change the Title to be something like "Sample Phone Printout".

Use **<CTL>-X** for the other values. This will keep the default output file named 'cdbprt'.

Your screen should look like the above when you are finished.

To exit the miscellaneous screen, use a **<RETURN>** at the Any Change prompt.

Note: the Print Detail item is used to control the printing of non-subtotal lines. When set to **no** only subtotals and totals are printed.

enter
data

Generating a Report

—

When you are satisfied with the items you have requested on the printout, enter the END key from the Any Change prompt.

Again, you will automatically be placed into the Miscellaneous screen on the way out. You will have to END from this screen also.

y

Then you will be asked the yes/no question: “Do you want to Generate Printout? (y/n)”. Since creating a report is the whole reason for this chapter, respond with a y here.

Or you can say go back to the menus, locate the printout generator tool in the MAIN menu.

Automatic Index Regeneration

The first thing that will happen (if needed) is the index will be regenerated. During this phase, the status area on the header line will say ‘Auto Regeneration’ and the middle of the screen will say ‘Working’.

When the index is regenerated, the report generator will move on to the next screen automatically.

Interactive Report Generation

When you are interactively generating a report, the printout generator will list all of the fields of this printout and display the miscellaneous dictionary, similar to the following sample from the Phone database:

```

FirstBase 6.5: dbpgen          FirstBase Software          Status: Parsing

dbpgen: Sample Phone Printout --> cdbprt.prt
      (spacing: 1; page width: 80; all detail)

Last_Name      a 35
First_Name     a 20
Initial        a 1
Telephone      a 15

Files: Phone(3/0) index(3/3) [john: /usr/tutorial 18:21 01/19/92]
If accurate, enter 'y' to continue : .

```

If a printout of the same name already exists, the printout generator will pause before this screen and request permission to destroy the old printout.

generation

This screen represents the generator's idea or 'picture' of what it thinks you want to generate. At this point, the generator will pause until it gets an answer from you.

If for some reason you do not want to generate the printout, answer the 'If accurate' prompt with a <RETURN>.

y

Once you respond **y** to the 'If accurate' prompt, the printout generator will fly through the database, its index and the printout dictionary you have defined, creating a report for you.

A counter appearing in the lower right hand quadrant of the terminal will display the number of records processed.

For each indexed record, the printout generator will add print lines to the printout for that record, according to your definitions in the printout dictionary.

reminder

You will get exactly the same number of processed records as there are entries in your **index**! If this index is only a selected portion of your database, then that is all that will be processed.

After the last line of the printout, a numeric count of the number of records processed is printed.

When the printout generator is done, you will be given the option of viewing and/or printing the report via the *screenprint*(8) tool. Merely follow the prompts if you want to see or print your report right now.

Afterwards, you will be returned to the *FirstBase* MAIN menu.

Congratulations — you have just created a *FirstBase* report.

This 'interactive mode' behavior is only the default. You can set *FirstBase* to run these generators without pausing for this display.

This silent approach is called batch processing — see the manual pages on *dbpgen*(1) and *generators*(5) for more details.

Viewing and Printing an Existing Report

All *FirstBase* reports are generated to disk storage. This means that you do not have to view/print your report immediately — you can always print the report again, without regenerating it.

If you did not change the name of the report (from the Miscellaneous screen described in this chapter), then the name of your report file will have defaulted to **cdbprt**.

tools

The printout generator added an extension to the file name, making the full file name **cdbprt.prt**. To see this file on your screen, go to the TOOLS menu by selecting **tools** from the *FirstBase* MAIN menu.

From this sub-menu, use the **screenprint** command.

You will be prompted for the name of the file to display. Your complete response to this prompt is **cdbprt.prt**.

screenprint

From this tool, which is actually *screenprint(8)*, you will be able to display the report to your terminal, or send it to your printer.

Here is a screen of the sample Phone report:

```
18:23 01/19/92                FirstBase Software                Page 1
                                Sample Phone Printout
Last_Name                      First_Name                      Initial
Telephone
Jackson                        Seymour                          P
@327-8221
Mills                          John                              J
@749-2864
Smith                          Fred                              B
@299-8771
*
@
3 records printed
~
cdbprt.prt END
```

More Complex FirstBase Reports

FirstBase provides many other tools for generating reports or output of some kind that can be used with other UNIX tools.

For instance, the tool *dbvform(1)* uses standard database view dictionaries and prints its result to the screen or to a file.

Another extremely flexible formatting tool is the database merge tool, *dbmerge(1)*. This tool, which can be used alone or as a preprocessor for UNIX tools like *nroff(1)*, is outlined in Chapter 10, "Document Merging".

There is also a mailing label generator tool. See Chapter 9, "Creating Mailing Labels".

Another report generator inside *FirstBase* is *dbsql(1)*. This tool will allow you to produce columnar output that has flexible headers and footers for both odd and even pages.

Still yet another tool that can be used for most anything, including a report generator, is *dbawk(1)*. This tool will allow standard *awk(1)* constructs to process databases. The output could be a report or data destined for some other UNIX tool.

Additionally, a *FirstBase* tool named *dbmacro(1)* can be used to process databases with a macro language, producing very fancy output.

Finally, *dbemit(1)* can be used to blast out comma separated values that can be piped into other text processing systems, shell scripts, or custom programs.

Again, the idea is tools. A few of these could be used as preprocessors to other UNIX tools, like *tbl(1)* or even *perl(1)*.

See the chapters and manual pages referenced for more details.

Summary

This chapter has discussed building simple printouts using the *FirstBase* tools *dbdpri(1)* and *dbpgen(1)*.

See these manual pages for more information on the more complicated features of the printout generator, such as break and total lines.

A section on more complex reports provided some pointers to other *FirstBase* tools that provide more flexible output.

*Ay me! what act,
That roars so loud, and thunders in the index?*

Hamlet
WILLIAM SHAKESPEARE

*You will find it a very good practice always to verify
your references, sir!*

DR. ROUTH

*“I quite agree with you,” said the Duchess; “and the moral of that is
— ‘Be what you would seem to be’ — or, if you’d like it put more
simply — ‘Never imagine yourself not to be otherwise than what it
might appear to others that what you were or might have been was
not otherwise than what you had been would have appeared to them
to be otherwise.’”*

Alice in Wonderland
LEWIS CARROL

For those who like this sort of thing, this is the sort of thing they like.

ABRAHAM LINCOLN

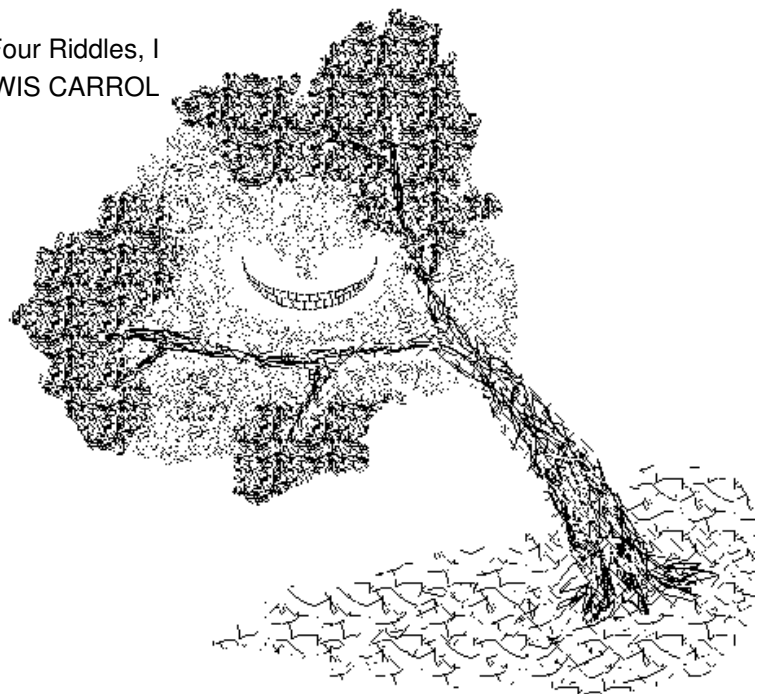
There are no answers, only cross references.

FORTUNE(1)

*Yet what are all such gaities to me
Whose thoughts are full of indices and surds?*

$$x^2 + 2x + 53 = \frac{11}{3}$$

Four Riddles, I
LEWIS CARROL



Creating an Index

The creation and development of *FirstBase* indexes is covered in this chapter. Indexes are one of the most powerful *FirstBase* data objects.

Mastering indexes and their uses will bring tremendous power to any *FirstBase* user. The *FirstBase* tools covered in this chapter are *dbdind(1)* and *dbigen(1)*.

Note: *dbsql(1)* can also generate standard *FirstBase* indexes using SQL constructs. See *dbsql(1)* for details.

What is an Index

basic
concepts

An index consists of particular fields of a database, created and organized in such a way as to quickly find the referenced record in the database.

The most simple indexes are made up of a single field.

Indexes must store a ‘pointer’ to the database record as well as the fields that are desired to be indexed. Furthermore, the index must be sorted to gain real power.

A sorted index can be searched quickly for a specific field value, and by following the associated ‘pointer’, the original record can be retrieved in very little time.

Many tools in *FirstBase* use indexes to provide structure and order for a database, as well as to ‘capture’ a selected set of records.

Any index generated can be used anywhere an index is allowed — regardless of the size and shape of the index.

An index can contain an entry for each record of a database, or a selected few.

Dbdind(1) is the formal mechanism for defining the index dictionary needed by *dbigen(1)* to generate the index.

Getting Started

ideas

The first thing to do is to decide what kind of structure or organization you want an index to impose on your database. If you want fast access to all of the records in an entire database, perhaps an automatic index will be sufficient.

If this is the case, you do not need to define or create an index. This is what we did in "AutoIndex the Database" on page 4-8 — we defined the LastName field of the database in such a way as to automatically maintain an index for us.

But, we cannot always keep multiple autoindexes — there is a machine dependent limit on how many autoindexes a database can have.

Furthermore, if the index is going to be used only once or twice, or if the index is more complicated than all records for a single field, then you will need to define and generate an index.

Creating the Index Definition

For now, let's create a very simple index for the Last Name field of the Phone database that is used as an example in the previous chapters of this manual.

Again, this is repetitious because we have defined this field to automatically maintain an index for us — but it will serve our purposes of example here.

Starting the Define Index Tool

The first thing you want to do is to name your database and index. See "Changing a File Name" on page 3-5.

If you are following the examples from this manual at your terminal, the database name, **Phone**, MUST be entered using this environment command screen. The example here depends on this name.

name
database
and index

Note that **Phone** has one capital letter: 'P'. The rest are lower case. *FirstBase*, and UNIX for that matter, **always** distinguish between upper and lower case.

You will want to name your index something more mnemonic than the default name also, such as **Lname**. This is accomplished in the same way as changing the database name.

Since we are just creating this index for the first time now, it does not exist — and will not show up in the status line as a valid index.

After creation of this index, any tool that references this index will have the index name 'Lname', displayed in the *FirstBase* footer line.

Do not let this absence of an index name confuse you — the names you have selected can still be seen by using the 'e' command from the *FirstBase* environment command screen.

1

Now, from the *FirstBase* MAIN menu, select the **Define Index** by entering the command **1** (one) and a <RETURN>.

The Define Index Screen

The following screen fragment shows what the top part of your screen will look like when you start the Define Index tool.

The idea is to build a list of selection criteria and items that the index generator will use to create the index using this screen:

```

FirstBase 6.5: dbdind          FirstBase Software          Status: Define Index
Field                          Val1 '>=' or $OP          Val2 '<=' or Field      And/Or
=====                      =====                      =====
  
```

A Selection Criterion

To add selection criteria for this simple index, go into the 'add mode' for this dictionary editor — in the same way as other *FirstBase* editors. Use the command '@' to put the editor into add mode.

@

The Field Column

The first screen you see after you begin add mode will be the standard Choose Field screen. This common screen is detailed in "FirstBase Choose Field Mechanism" on page 2-12.

1

For this example, enter a 1 (one) to indicate 'Last Name'.

The Val1 Column

After you choose from the list of field names, another choice screen will be presented, allowing you to select choices for the Val1 column:

```

FirstBase 6.5: dbdind          FirstBase Software          Status: Choose Value

                               Available choices for Val1 '>=' or $OP

Choice   Meaning Comment
-----
Use these to control the comparisons:
all      $ALL   Gather all records, no comparisons. (default).
not      $NOT   To gather all records Except some set.
empty    $EMPTY Gather records with empty field.

Use these to compare with other Fields:
eq       $EQ    Equal
ge       $GE    Greater Than or Equal
gt       $GT    Greater Than
le       $LE    Less Than or Equal
lt       $LT    Less Than
ne       $NE    Not Equal

Or use 's' to enter a '>=' Value (Val1):
s        $STRING

Files: Phone(3/0) Lname(0/0) [john: /usr/tutorial 23:31 01/19/92]
Enter choice: . Select Choice
  
```

The value that you enter here, the **Val1** column, will be used only on its corresponding field name.

Often times, when only a few selected records is the desired goal, you will enter a sequence of characters here that will be used, along with the next column, to select a range of records.

For example, to start matching records with Last Name of **Jones**, you would enter an **s**, and the string 'Jones' at the Enter String prompt.

Similar selections can be made for the **Val2** column.

all

But, in this simple example, we want to capture all the records. So, use **all**, which will plug in the meta value '\$ALL' to select all of the records.

Actually, the default for the **Val1** column is '\$ALL'. So, a <RETURN> at this point will do the trick as well.

When all records are requested using **\$ALL**, there is no need to enter a second value for the **Val2** column — all records of the database are being selected regardless of what value they have stored in their Last Name field.

Also, since this is the first selection criterion (and the only one for this very simple index), the last column, the **And/Or** column is not prompted for either. This column is used to tie together two or more selection criteria — as we will do in a later example.

Finishing One Entry

—

Now you will be asked if you want to Add More Entries. For this simple example, the response is **n** or **END**.

If you accidentally go on to the choice screen for line 2 of the index, use the **END** keystroke to get back to the Any Change screen.

At this point you should get a display similar to this fragment:

```

FirstBase 6.5: dbdind          FirstBase Software          Status: Choose Value
Field                          Val1 '>=' or $OP            Val2 '<=' or Field          And/Or
====                          =====                    =====                    =====
1) Last Name                    $ALL

```

Again, this simple selection criterion means that when the index generator is run, all records that have a Last Name of any value will be selected. In other words, all records of the entire database.

basic
concepts

The 'Sort By' Screen

Now that we have a selection criterion in place, we can decide exactly what it is we want the index to use as the reference field(s) — the fields that will be written to the index for sorting.

These fields are called the **Sort By** fields.

Do not get confused — the selection criteria govern only what records we want the generator to select or capture.

Once a record is selected, only fields that we list at this 'Sort By' screen will be physically stored in the index, along with a 'pointer' to the record.

After the index is completely written, it is sorted.

So, if the **Sort By** column is the entry that controls the actual sorted order of the records, then why did we use 'Last Name' in the selection criterion in "The Field Column" on page 8-3?

So as not to produce too much confusion at one time.

Actually, any valid field could have been used, with the special value '\$ALL', and the result would be the same. But, let's not get bogged down with too many details all at once.

The first time you define an index, as you END from the Any Change prompt, you will be placed in the **Sort By** screen automatically.

Or, you can get to the 'Sort By' screen by using the **s** command from the Any Change prompt. This will produce a screen that looks like:

```

FirstBase 6.5: dbdind          FirstBase Software          Status: Choose Value
Selected 'Sort By' Fields
=====
      (by: Last Name)
  
```

Note that it already has what we really want the index sorted by — Last Name. This is because the default for the field(s) to sort by is the first field of the database.

If you did want to make changes here, you would enter a **y** and a <RETURN>. This action would invoke a fairly standard Choose Field screen, covered in "FirstBase Choose Field Mechanism" on page 2-12.

The difference is that you can choose more than a single field. Either one at a time, or a few per line, separated by blanks. The fields selected are the actual **Sort By** fields.

For example, if you were inside the **Sort By** screen and had responded with a **y** at the **Any Change** prompt, the **Phone/Lname** database example would display a screen like:

```
FirstBase 6.5: dbdind          FirstBase Software          Status: Choose Value

      1> Last Name  a          35
      2> First Name a          20 [ ]
      3> Initial   a           1
      4> Telephone a          15

Files: Phone(3/0) Lname(0/0) [john: /usr/tutorial 23:38 01/19/92]
Enter SortBy #'s: |.....                               -=End
```

At this point, you could enter the number '1' to use the Last Name field as the only sorted field.

If you wanted to sort and reference a database by Last Name and First Name, you could enter '1 2' here to get both fields number 1 and number 2.

Alternately, you could enter a '1' (and a <RETURN>), then a '2' (and its <RETURN>), since these values are all saved or stacked up.

When you are done selecting **Sort By** fields, use the **END** keystroke to exit to the **Any Change** prompt. Then, use the **END** here, also.

Again, the fields on the '**Sort By**' screen are the ones that are going to be written to the index, the field values that will be sorted.

These indexes can then be used with many tools in *FirstBase*, including database editors and report generators.

Note: you cannot use fields containing multiple text lines in them as **Sort By** fields. In other words, don't sort by *FirstBase* visual fields, those fields edited using an external UNIX editor.

Generating an Index

Now comes the easy part — generating the index.

If you have defined an index or gone into the Define Index program, then when you exit you will be asked whether you want to run the index generator.

y

Respond with a **y**, and the Index Generator will be run for the index you have just defined.

Or, you may have set up the dictionary at some earlier time and need to go to the Index Generator tool without going through the Define Index tool. This can be done through the *FirstBase* MAIN menu, also.

7

Again, the names of the database and index need to be set correctly. Then from the MAIN menu, choose Generate Index, selection **7**.

No matter how you start the index generator, you will get the same screens and results — assuming that the same database and index names are used.

generation

Per the normal *FirstBase* generator method, a ‘picture’ of what the generator thinks you want to generate will be displayed.

For example, the index generator screen using the simple Lname index will look like:

```

FirstBase 6.5: dbigen      FirstBase Software      Status: Lname
{
  (Last Name: $ALL )
}
(by: Last Name)

```

y

When you respond with a **y** to the ‘If accurate’ prompt, the generator will be off and running.

The index generator will then go through a couple of steps to complete its task — selecting, and then sorting. In both cases, the status area will be updated to reflect what stage the generator is working on.

<RETURN>

Once done, the number of selected records is displayed, and you will need to hit a **<RETURN>** to return to the menu.

That’s it — the index has been generated and can now be used for printouts, labels, document merging, and many other things.

Even the index generator can use an existing index to work from when generating a new index.

Indexes are building blocks — play with them.

This ‘interactive mode’ behavior is only the default. You can set *FirstBase* to run these generators without pausing for the user.

This silent approach is called batch processing — see the manual pages on *dbigen(1)* and *generators(5)* for more details.

Another Example

Now that you are fully index literate, we can move on to an example that is a bit more interesting.

Let’s say you want to capture all records in your existing database that have a Last Name somewhere between ‘Jackman’ and ‘Jacobs’ who also have a middle initial of ‘P’. A bit more tricky, but not that complicated.

The idea is to formulate the selection criteria so they match the desired request.

You can use the same index name if you want, but remember that you are destroying an existing index.

hints

These details are left to you — but one suggestion: it is much easier to locate indexes (and UNIX files in general) if they are named something that reminds you of their content.

In any case, you will start up the Define Index tool, and go into add mode to create selection criteria.

getting started

Or, if the index already exists, you may just be modifying existing criteria.

In this case, you would use the same editing skills you learned for the database editor of Chapter 4 and Chapter 5 to make the modifications to the selected dictionary lines.

reminder

Modifications are done by entering the number of the item you want to change at the Any Change screen (and a <RETURN>).

Remember this editing method?

Modify any columns you want, and use <CONTROL>-X to skip across columns or choices you want to leave intact.

For this example, we will modify the Lname index too look like:

Field	Val1 '>=' or \$OP	Val2 '<=' or Field	And/Or
1) Last Name	Jackman	Jacobs	\$AND
2) Initial	P	\$NONE	

s

To make the above changes to the Lname index, use the **s** command during the selection of the Val1 column, from the screen shown in "The Val1 Column" on page 8-3. Then enter the value 'Jackman'.

This time (since \$ALL was not selected) you *will* be presented with the Val2 column screen.

The Val2 Column

The Val2 column is used to enter a second value that is used in conjunction with the Val1 entry to trap values of fields within a certain range.

There are also some additional choices here, such as \$NONE, that will dictate special behavior between the Val1 and Val2 entries.

Here is what the Val2 selection screen looks like:

Choice	Meaning	Comment
Use these to control how Val1 is used:		
none	\$NONE	Gather records equal to Val1 Constant. (default).
pat	\$PATTERN	Gather records matching Regular Expression Val1.
empty	\$EMPTY	Use with \$NOT as Val1 to catch non-empty fields.
Or use 's' to enter a '<=' Value (Val2):		
s	\$STRING	

s

Here again, you will use the **s** command for the Last Name selection so as to enter a string value of 'Jacobs', indicating a range of values.

none

However, for the Initial field we are going to use the special Val2 connector named \$NONE. This special value means that only records with *exactly* the Val1 value will be selected (by that portion of the selection criteria).

When you are done with the Val2 entry for the Initial field, you will be given a screen to edit the And/Or column.

Logical Connectors - And/Or Column

After you are done entering the Val2 column selection for the Initial field from the Lname index example, you will be presented with the And/Or column.

basic
concepts

Two or more selection criteria items are bound together by a logical connector — either an ‘AND’ or an ‘OR’ connection.

By placing many selection criteria together and binding them with logical connections, a very selective index can be created.

The following is the And/Or column screen:

```

FirstBase 6.5: dbdind          FirstBase Software          Status: Choose Value

                Available choices for And/Or Connector

Choice   Meaning Comment
-----
Use these to control how the surrounding requests
will be connected together:

a        a        AND. Both Must be true to satisfy. (default).
o        o        OR.  Either one Must be true to satisfy.
s        s        Simple And. Used to connect two conditions
                inside of surrounding Or requests.
  
```

a

In our example here, we are using the AND connector, selected by using the **a** command from this screen. This connector means that only records that satisfy *both* criteria are actually selected.

The behavior of the AND connector is much different than the OR connector, which would only require one OR the other (or both) of the criteria to hold true for a record to be selected.

more
concepts

Of the possible logical connectors, the AND has a higher precedence, meaning it is stronger than any OR connections.

However, there is a SIMPLE AND connector that can be used in the midst of OR connectors since it has a lower precedence.

For a very complicated series of criteria and logical connectors, it may help to see a different ‘picture’ of the selection items.

The Define Index tool provides a trace command, **t**, which will produce an indented, parenthesized picture of the selection items.

Often, this tracing will help visualize the ‘selection structure’.

Completing the Example

generation

After selecting what fields you want to Sort By, you can generate this semi-interesting index from the end of the Define Index tool, or from the Generate Index selection in the MAIN menu.

Using the data as depicted in "The Database Revisited" on page 5-1, only one record, Record 2, will be selected. This record is the only one that fits all of the selection criteria.

A Word on AutoIndexes

hints

Autoindexes are special *FirstBase* indexes that are maintained during record insertion and deletion. For normal or flat indexes, standard index dictionaries are created by the editors as needed. These dictionaries should not be changed—they must be simple, single field, \$ALL selections.

Another type of *FirstBase* index is the Btree index. A Btree index can also be defined as an autoindex. If the index is sorted by a single field, it can be a defined autoindex in the database dictionary (as long as AUTOBTREE is ON via *setup(5)*).

If you want to maintain a complex, multi-valued index, you will have to create the index dictionary and store the name of it in the autoindex file as explained in *autoindex(5)*. Only Btree indexes can be used as an autoindex in this manner.

Also, if you are having trouble aligning an index with a database, use *rmdir(8)* and regenerate the index, or use *dbigen(1)* with a **-f** flag.

You can also rename or move an index using *mvidx(8)*. If you move an autoindex, remember to change the name in the database dictionary or autoindex file also.

Index Internals

There are two types of *FirstBase* indexes: flat or normal and Btree. Each type has their advantages, but the newer Btree mechanism is probably the index type of choice for autoindexes that change often.

Normal Indexes

A normal *FirstBase* index is stored on disk as a list of the Sort By fields aligned in such a way as to utilize the UNIX *sort(1)* facility. Each element of the list also contains a record number, or pointer, to the actual record in the database file.

A separate file, the index header file, is used to store the Index Record Count, the number of records stored in the index.

Additionally, the index header file is used to store the Index Organiza-

tion Count, a number that represents the point through which the index is organized, or sorted.

Any index entries *past* this point are not organized (not sorted) at all.

In generalized *FirstBase* indexes, the Record Count will match the Organization Count exactly. However, within Autoindexes the Record Count can extend beyond the Organization Count.

This area of an index between the Organization Count and the Record Count is called the Overflow Area.

The Overflow Area of a *FirstBase* index is searched sequentially by the database editors after doing a binary search on the organized (sorted) area of the index.

This Record Count, Organization Count and Overflow Area all mean that Autoindexes can become out of date, or at least appear to be unsorted.

A special *FirstBase* tool named *dbregen(1)* will sort an index and update the index header file without rewriting the index from the database.

Btree Indexes

Btree indexes within *FirstBase* provide a mechanism that maintains the order of the index through all record additions, deletions, and modifications.

With Btree indexes (that are autoindexes), the data represented is always in sorted order and never needs to be regenerated.

A Btree index has a few components: the index dictionary, the index header file, the btree sequence set and the btree index set. The index header contains the Record Count, much like normal indexes.

The other two components together, the sequence set and the index set, represent the selected records (all records for autoindexes). These files are stored using an internal tree structure of linked nodes and are *never* used with any other UNIX utilities like *sort* or *grep*.

When defining an index, the type of the index-to-be is controlled by a flag within the index dictionary. Although *FirstBase* assumes you want a Btree index, the commands **tree** and **norm** can be used to toggle the index type. The define index tool (*dbdind*) displays the index type in the lower right corner.

Again, Btree indexes, when used as autoindexes, provide the advantage of always being in sorted order. These indexes provide both a sorted (sequence) set of records as well as very fast random access to any individual record.

Summary

This chapter has discussed the simpler aspects of creating an index using the index dictionary editor and the index generator. Still more details are described in the manual pages on *dbdind(1)* and *dbigen(1)*. See *autoindex(5)* for discussion of complex autoindex maintenance.

Additionally, note that *dbsql(1)* can also generate normal *FirstBase* indexes using SQL constructs. See *dbsql(1)* for details.

There is even a tool that will take an existing index and filter it according to a list of patterns, creating a new index. See *dbfilter(1)*.

UNIX
Note

For all the UNIX oriented people: an index dictionary is a text file that can be generated by other UNIX tools if needed.

Use of a text editor or other UNIX tools on any of the standard *FirstBase* dictionaries is permissible and is a feature, but should not be done by the casual user — in any case.

The relevant manual pages are *dictionaries(5)* and *idicti(5)*.

The material presented here discusses the *FirstBase* full screen database screen dictionary editor, *dbdind(1)*, which does complete syntax checking for the data dictionary.

If you are still confused, stick to simple indexes for a while. The more complicated indexes will become easy as you learn *FirstBase*.

Some users of *FirstBase* have not realized the full potential power of the indexing tools until well after they have created applications.

Indexes, much like printouts, can be easily redone at any time. Do not be scared to try something — you can always regenerate the index.

Learn the mechanisms — your *FirstBase* knowledge will grow in time.mm



*I will give out divers schedules of my beauty: it shall be inventoried,
and every particle and utensil labelled to my will; as Item, Two lips,
indifferent red; Item, Two grey eyes with lids to them; Item, One neck,
one chin, and so forth.*

Olivia
Twelfth Night
WILLIAM SHAKESPEARE

*I have fallen in love with American names,
The sharp names that never get fat
The snakeskin-titles of mining claims,
The plumed war-bonnet of Medicine Hat,
Tucson and Deadwood and Lost Mule Flat.*

STEPHEN VINCENT BENÉT

*For all a rhetorician's rules'
Teach nothing but to name his tools.*

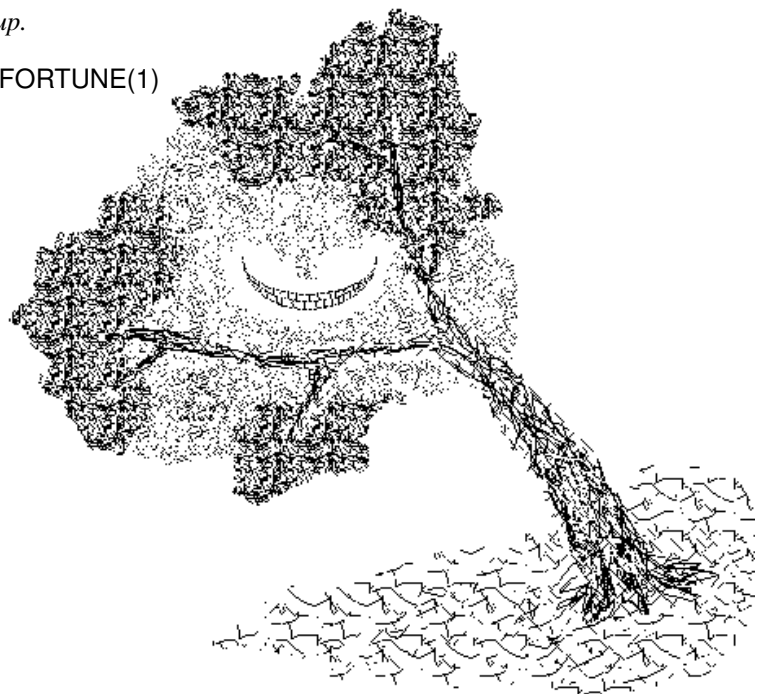
SAMUAL BUTLER

*The naming of Cats is a difficult matter;
It isn't just one of your holiday games;
At first you may think I'm mad as a hatter
When I tell you a cat must have THREE DIFFERENT
NAMES.*

The Naming of Cats
T. S. ELIOT

*Anything labeled "NEW" and/or "IMPROVED" isn't. The label
means the price went up. The label "ALL NEW", "COMPLETELY
NEW", or "GREAT NEW" means the price went way up.*

FORTUNE(1)



Creating Mailing Labels

The definition and generation of mailing labels is covered in this chapter. A slight understanding of *FirstBase* indexes is assumed. The *FirstBase* tools covered here are *dbdbl(1)* and *dblgen(1)*.

Mailing Labels

FirstBase provides tools that allow the easy design and generation of mailing labels. The labels that are produced can be printed on peel off labels or ready to cut paper.

Getting Started

preliminary
assumptions

The first thing to do is to make sure that all the fields you will want in your mailing labels are well defined in your database.

You will also need to define an index for the label generator to run from. This index will determine which database records are processed into labels, as well as the order they are processed in.

Furthermore, we will assume that other fields exist in this sample Phone database that we have not previously seen — things like Home Address, City, State, etc.

reminder

You will get exactly the same number of processed records as there are entries in your **index**! If this index is only a selected portion of your database, then that is all that will be processed.

Creating the Label Definition

name
database
and index

For now, let's create some simple labels using the Phone database we have provided for examples in previous chapters.

Starting the Define Label Tool

The first thing you want to do is to name your database and index. This can be done by using the environment command screen described in "Changing a File Name" on page 3-5.

Phone

If you are following the examples from this manual at your terminal, the database name, **Phone**, MUST be set using this environment command screen. This is a must — otherwise your database will be named something different.

The name of the label dictionary will be produced by using the name of the index — you do not have to think of another file name.

Lname

Make sure that you are using the index name you want the label dictionary to be associated with.

For our example here, let's use **Lname**. The index name is defined using the same method as the database name.

3

Now, from the *FirstBase* MAIN menu choose the selection titled **Define Labels** (number 3) to use the *FirstBase* tool for defining a label dictionary.

The Define Label Screen

The following screen shows what your screen will look like when you start the Define Label tool:

```

FirstBase 6.5: dbd1b1          FirstBase Software          Status: Define Labels
/-----\
0>
1> [ ].....
2>
3>
4>
\-----/
5> Output to: ---> cdb1b1

Label Format:  4 up
Label Size:   54 char/label

Left Justify 0? n
Files: Phone(3/0) Lname(3/3) [john: /usr/tutorial 11:35 01/20/92]
Enter Label Line, <CTL>-C = abort

```

**basic
concepts**

The idea is to insert the particular field names that you want to appear in the label into a five line label template.

The label generator will use this label design to create the labels.

The Label Template

The label template consists of 5 blank lines that will be used to create a dictionary for custom mailing labels.

The cursor will be positioned on the line numbered one. (The line numbered zero is usually not used, although you can come back to it later).

Unlike most other *FirstBase* dictionary editors, the Define Label tool begins in a kind of add mode — taking you from line one through line five.

At the end, you will be allowed to make changes to any of the lines entered, plus the 0th line and the miscellaneous label information.

For each line, enter field names, characters and punctuation that you want to appear in the label.

If you have embedded blanks in your field names, substitute an underscore for each blank when you are using that field.

For example, on the first line put both the First Name and the Last Name fields, like a proper mailing address.

To do this, you will enter the characters 'First_Name Last_Name', and a <RETURN>.

The Define Label tool will break each item you enter into a 'token', and decide whether that token or group of characters is defined in the database dictionary or not.

If you enter a series of characters that is not defined, the Define Label tool will notify you with a simple up arrow pointer (caret) that will point at the first character of the undefined token.

Notice in the finished label template that the 'or', the 'Occupant' and the 'at' *tokens* are all undefined.

Undefined tokens will simply be output as they appear.

Again, we went back and added four more fields to the Phone database so that the mailing label example shown here would be somewhat more realistic.

But, you do not need to do this — you just won't generate quite the same screen as listed here. The new fields added to the Phone database are Address, City, State and ZipCode. If these fields are not defined, then they too will be marked as undefined tokens.

field
names

undefined
tokens

punctuation

You can intersperse your field names with punctuation (periods, commas, etc.). The Define Label tool will separate out punctuation so that you can use something like 'State.' where only the State is defined — the period will be printed in the labels at label generation time.

The Define Label tool understands punctuation, and does not bother to flag these characters as 'Undefined'.

Most mailing labels can be defined on just three lines — but a fourth is provided also. Furthermore, the top line, which is meant to contain an identifying code or batch number for the labels, can really be used for the first line of the label if desired.

The following screen is a completed label template:

```

FirstBase 6.5: dbdlbl1          FirstBase Software          Status: Define Labels

      /-----\
      | 0>                                           |
      | 1> First_Name Last_Name or Occupant at      |
      |      ^      ^      ^                        |
      | 2> Address                                  |
      | 3> City, State. ZipCode                     |
      | 4>                                           |
      | \-----/                                   |
      | 5> Output to: ---> cdblbl1                  |
      |                                             |
      | Label Format:  4 up                          |
      |                                             |
      | Label Size:   54 char/label                 |
      |                                             |
      | Left Justify 0? n                           |
      |-----|-----|-----|-----|-----|
      | Files: Phone(3/0) Lname(3/3) [john: /usr/tutorial 11:39 01/20/92] |
      | Any Change (#, <CTL>-H=Help, -=End) ? [.]... |
  
```

cdblbl.lbl

Miscellaneous Label Information

The Define Label tool also provides a place to input the name of a file where the print image of the labels will be stored. This name defaults to 'cdblbl.lbl'.

Note that the Label Generator will always add an extension to the filename of '.lbl'.

Other information stored in the area below the label template includes the label format and the label size.

format

The label format designates how many labels you want to appear across the top of the output.

The most common formats seem to be one and four, but anything in between will also work.

size

The label size is the number of characters in width each label will have.

You may need to play with this number and your own labels to find what the correct number is.

We use a label size of 56, which works well for us when the labels are printed using 12 characters per inch (pitch).

One way to determine the label size you need is to create a file using your text editor (like *vi*(1)), and put '0123456789....' on a few lines, carried out to at least column 50 or 60.

Then print that file on top of some of the labels you will be using.

Make sure to use the same pitch setting on the printer as you will want to use when you actually print the labels later.

Making Corrections

Like all other *FirstBase* tools, you can make corrections to any of the fields before you exit from the *FirstBase* tool.

These are accomplished by typing the number of the line you want to change at the Any Change screen.

making
corrections

Once the cursor is on the line you want to change, a <CTL>-X will move you across columns without disturbing the data that is there.

You can continue to make corrections to the label template and the other information until you are done.

Once in place, it will always be there if you need to come back and slightly modify the definition again.

—

When you are done with the template, exit the Define Label tool using the standard END key '-'.

You will be given the option of running the Label Generator.

If you do not run it now, you can run it from the MAIN menu later.

Generating Labels

Now comes the easy part — generating the labels.

If you have defined the labels or gone into the Define Label program, then when you exit you will be asked whether you want to run the label generator.

y

If you respond with a y, the Label Generator will be run for the labels you have just defined.

Or you can say go back to the menus, locate the Label Generator tool in the MAIN menu.

No matter how you start the Label Generator, you will get the same screens and results — assuming that the same database and index names are used.

Automatic Index Regeneration

The first thing that will happen (if needed) is the index will be regenerated. During this phase, the status area on the header line will say ‘Auto Regeneration’ and the middle of the screen will say ‘Working’.

When the index is regenerated, the generator will move on to the next screen automatically.

Interactive Generation

When you are interactively generating labels, the generator will display a listing of the label along with the miscellaneous information.

generation

This screen represents the generator's idea or ‘picture’ of what it thinks you want to generate.

At this point, the generator will pause, waiting for an answer.

If for some reason you do not want to generate the labels, answer the ‘If accurate’ prompt with a <RETURN>.

y

Once you respond y to the ‘If accurate’ prompt, the Label Generator will fly through the database, its index and the label dictionary you have defined, creating labels for you.

A counter appearing in the lower right hand quadrant of the terminal will display the number of records processed.

For each indexed record, the Label Generator will create a single label.

The last label generated will display a numeric count of the number of labels generated.

When the Label Generator is done, you will be given the option of view-

ing and/or printing the labels via the *screenprint*(8) tool. Merely follow the prompts if you want to see or print your report right now.

Afterwards, you will be returned to the *FirstBase* MAIN menu.

Viewing and Printing Existing Labels

To view existing labels, follow the directions in "Viewing and Printing an Existing Report" on page 7-7, only use **cdlbl.lbl** or the name you used in this chapter, under "Miscellaneous Label Information" on page 9-4.

Summary

This chapter has discussed the more simple aspects of creating custom mailing labels using the label dictionary editor and the label generator.

Still more details are described in the manual pages on *dblbl*(1) and *dblgen*(1).

Mailing labels can be used for other purposes also. Conference badges or magnetic tape labels could be produced in the same manner.

The section named "More Complex FirstBase Reports" on page 7-8 might also be of interest for other ideas on creating more complex labels.

*The fountains mingle with the river,
And the rivers with the ocean;
The winds of heaven mix for ever
With a sweet emotion;
Nothing in the world is single;
All things by a law divine,
In one another's being mingle.
Why not I with thine.*

PERCY BYSSHE SHELLEY

*Mix a little foolishness with your serious plans; it's
lovely to be silly at the right moment.*

HORACE

*When bad men combine, the good must associate; else they will fall,
one by one, an unpitied sacrifice in a contemptible struggle.*

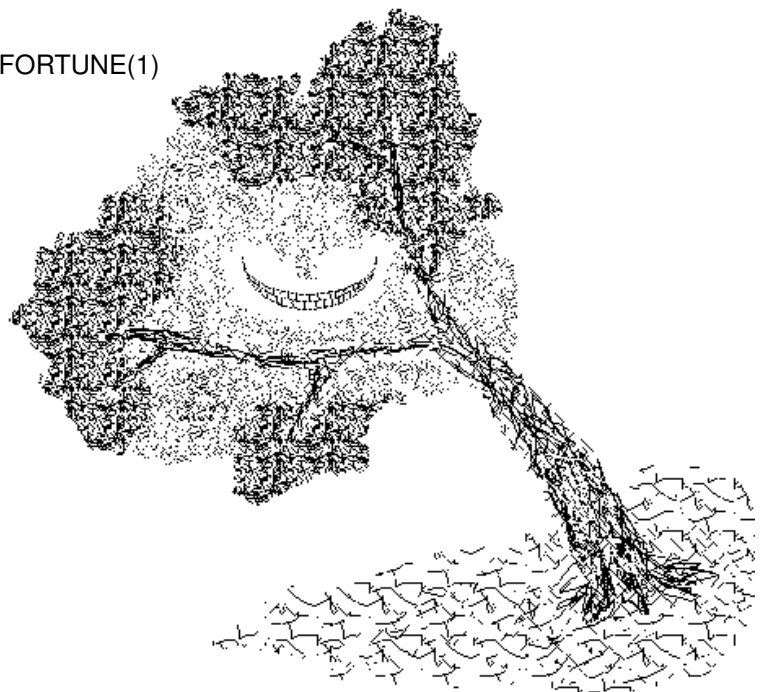
EDMUND BURKE

*[When asked with what he mixed his colours.]
I mix them with my brains, sir.*

JOHN OPIE

*Behold the warranty... the bold print giveth
and the fine print taketh away.*

FORTUNE(1)



Document Merging

The free form merging of *FirstBase* databases and text is discussed in this chapter.

A slight understanding of *FirstBase* indexes is assumed.

You can use your own editor, like *vi*(1), to edit the input document, or use *dbdmrg*(1), a *FirstBase* tool which provides full screen editing of a merge document using *vi*(1) style keystrokes.

The *FirstBase* tool covered in this chapter is *dbmerge*(1).

Merging Databases with Text Files

basic
concepts

FirstBase provides a tool that allows the merging of edited textual matter with database records.

The output of this tool can be used as input to other standard UNIX formatting programs such as *nroff*(1).

Alternately, if you use *dbdmrg*(1) to build the document, you can produce formatted output directly from the document merge tool.

This tool, *dbmerge*(1), effectively implements a ‘mail merge’ program.

Most often, *dbmerge*(1) is used to create a copy of a document for each indexed record of a database, where each document has some variable fields in it.

For example, if you need to send the same letter to many people, with the only difference being an address or an amount due, you can create a single copy of the letter and merge the letter with selected records of a database.

The output would be a letter for each record (or person), with the information specific to that record plugged into the proper places in the letter.

Thus, *dbmerge*(1) can be used for invoices, check writing, mass mailings, or any kind of database/text merging.

reminder

You will get exactly the same number of processed records as there are entries in your **index**! If this index is only a selected portion of your database, then that is all that will be processed.

Getting Started

getting
started

The first thing to do is to create a text file using *dbdmrg(1)* or your favorite editor.

Either will work, though *dbdmrg(1)* will compensate for field width and display a proper merge image.

See *dbdmrg(1)* for details on its use.

If you plan on using *nroff(1)* — which works well — create the *nroff(1)* source file as you normally would, interspersing text with formatting commands.

In this way, *dbmerge(1)* is actually a pre-processor for the *nroff(1)* source — i.e., the output of *dbmerge(1)* can be fed as input to *nroff(1)*.

Dealing with a text editor is beyond the scope of this chapter.

However, you can get to UNIX from the menu environment screen, as seen in "External UNIX Shell" on page 3-6.

There are vast amounts of documentation on editing documents. See your system manager if you are unfamiliar with any of this.

Creating the Merge Source

edit
source
document

In any case, when creating your *dbmerge* source document, any fields that you want to be substituted into the text at merge time are prepended with a dollar sign.

Also, embedded blanks need to be replaced with an underscore.

For example, to use the field 'Last Name' in your document, put the characters '\$Last_Name' in the text.

Dbmerge(1) will substitute the value of the field for this 'token' or series of characters.

For letters, you could put a separate salutation for each letter. For invoices, you could put the dollar amount due there.

Fields can also be printed in constant width.

Dbdmrg(1) will prompt for this value. However, if you are using your own editor, the syntax is '\$Last_Name[35]'.

You can still use the dollar sign in the text without it being expanded.

For example, the token '\$100.00' does not represent a valid field name and will wash through the merge cycle untouched. Also, two dollar signs can be used to print a single dollar sign.

Running the Document Merge Tool

Now comes the easy part — merging the source with the database.

If you have used your own editor, you can start the document merge tool using from the MAIN menu.

name
database
and index

Remember to name your database and index using the methods from "Changing a File Name" on page 3-5.

If you are using `dbdmrg(1)`, you will be able to run the document merge tool directly from within a sub screen.

No matter how you start the Document Merge Tool, you will get the same screens and results — assuming that the same database and index names are used.

Automatic Index Regeneration

The first thing that will happen (if needed) is the index will be regenerated. During this phase, the status area on the header line will say 'Auto Regeneration' and the middle of the screen will say 'Working'.

When the index is regenerated, the generator will move on to the next screen automatically.

Interactive Generation

When you are interactively merging documents, the generator will display a listing of the 'dollar tokens' that were found when the source was read.

This screen represents the generator's idea or 'picture' of what it thinks you want to generate.

At this point, the generator will pause, waiting for an answer.

If for some reason you do not want to generate the merge document, answer the 'If accurate' prompt with a `<RETURN>`.

enter
input and
output
file names

Once you respond `y` to the 'If accurate' prompt, the Document Merge generator will ask you for a couple of pieces of information: the **input** file name and the **output** file name.

After these file names are entered, `dbmerge(1)` will go through the database, its index and the label dictionary you have defined, merging your source with selected data.

A counter appearing in the lower right hand quadrant of the terminal will display the number of records processed.

generation

For each indexed record in the database and index you are using, `dbmerge(1)` will create exactly one copy of the source file with all the valid database field values substituted in.

Each copy is concatenated together (placed end to end) to create a single output file.

When the merge generator is done, you will be given the option of viewing and/or printing the report via the *screenprint(8)* tool. Merely follow the prompts if you want to see or print your document right now.

Afterwards, you will be returned to the *FirstBase* MAIN menu.

Since all the copies of the source are placed one after the other in the output, you will probably want to have some kind of separator.

For some cases, i.e. very simple database reports, maybe a 'new-line' character will be enough. For more complicated record/text merging, dashed lines might be appropriate.

If you are going to use the output of *dbmerge(1)* as input to *nroff(1)*, perhaps you want to begin each new record on a new page.

This would fit with the 'mail merge' analogy. In this case, place the *nroff(1)* command '.bp' either at the top or the bottom of your merge **source** document.

Viewing and Printing Merged Document

To view existing merged documents, follow the directions in "Viewing and Printing an Existing Report" on page 7-7, only use the file name you used as the output file here

Summary

This chapter has discussed the more simple aspects of merging documents with a database using the Document Merge program.

Still more details are described in the manual pages for *dbdmrg(1)* and *dbmerge(1)*.

This single tool provides tremendous power for *FirstBase*. Very customized reports can be created and 'generated'.

The section named "More Complex FirstBase Reports" on page 7-8 might also be of interest for other ideas on creating more complex documents.

*God! I will pack, and take a train,
And get me to England once again.*

RUPERT BROOKE

Now mark me how I will undo myself.

Richard III
WILLIAM SHAKESPEAR

*Don't be frightened; I am recalled. Pay, pack, and
follow at convenience.*

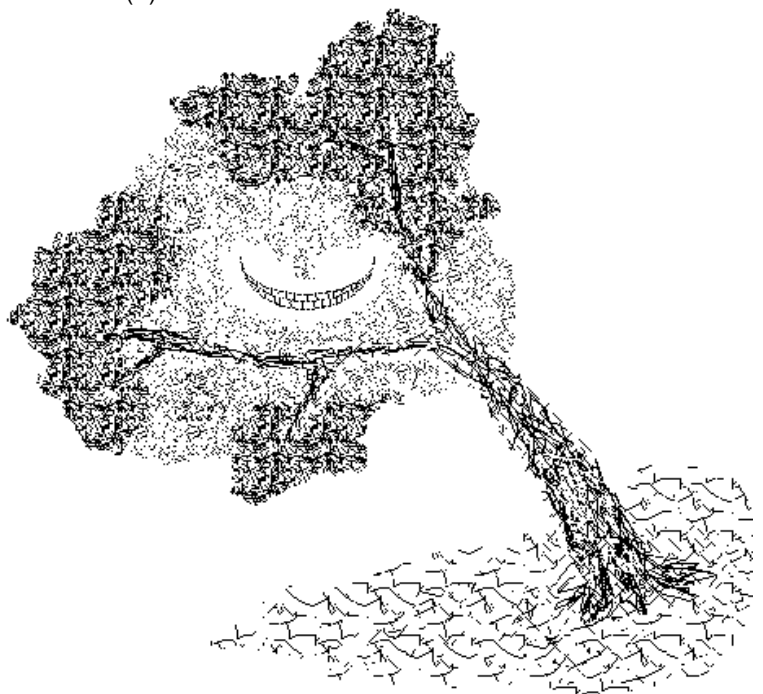
SIR RICHARD BURTON

*To see and be seen, in heaps they run;
Some to undo, some to be undone.*

JOHN DRYDEN

Think of it! With VLSI we can pack 100 ENIACs in 1 sq. cm.!

FORTUNE(1)



Removing/Restoring Deleted Records

The removal and restoring of database records is discussed in this chapter. Indexes do not have any effect on these tools.

The *FirstBase* tools covered here are *dbpack*(1) and *dbundo*(1).

Deleted Database Records

In Chapter 5, the database editor, we learned how to delete a record. (See "Deleting The Record: 'del'" on page 5-12).

Again, deleting a record using the database editor, *dbedit*(1), is considered a modification to the record since it is not really removed from the database.

basic
concepts

To physically remove the record from the database file, a special *FirstBase* tool is used. The process is called 'packing a database' — and the *FirstBase* tool is named *dbpack*(1).

Restoring a deleted record can also be done using *dbedit*(1), although this can only be done for one record at a time.

FirstBase provides a tool that will undo all of the deletions in a database in a single action. This tool is named *dbundo*(1).

In reality, the packing and undoing of a database is so very similar that these tools are really one in the same — the action done depends on how the program is invoked.

(For you UNIX people, *dbpack*(1) is linked to *dbundo*(1) — the name of the program is used to decide what to do!)

Since the actions of these tools are so very similar, we will discuss them together.

Getting Started

As mentioned above, indexes take no part in either packing or undoing a database — in both cases, the entire database is processed.

Furthermore, after either of these tools is used, you will need to regenerate all indexes associated with that database that are not autoindexes.

This regeneration of indexes is needed because the database is either rewritten, or records that might not be in some newer indexes are restored.

name
database
and index

From the menus, you will need to make sure the database is named properly. See "Changing a File Name" on page 3-5.

The menu selections for removing or restoring deletions are located in the *FirstBase* TOOLS menu.

These selections are labeled **pack** and **undo** respectively.

Packing and Undoing

pack

Both database packing and undoing process an entire *FirstBase* database.

Once started, the user is given a chance to back out before the process actually begins.

undo

A counter on the screen indicates which record is being processed.

At the completion of the process, the database is automatically 'cleaned' (using the *FirstBase* tool *dbclean(1)*).

This post processing is done in batch mode from the original tool (*db-pack(1)* or *dbundo(1)*), although the screen does indicate what is occurring.

See the Chapter on Manager Tasks and the man page on *dbclean(1)* if you are hungry for details on what cleaning does.

Summary

This chapter has discussed removing and restoring deleted database records.

There is really not much to these tools — they do as you would expect.

However, a few more details are described in the manual pages on *db-pack(1)* and *dbundo(1)*.

The Queen was in a furious passion, and went stamping about, and shouting 'Off with his head!' or 'Off with her head!' about once a minute.

Alice's Adventures in Wonderland
LEWIS CARROL

*Against that time when thou shalt strangely pass,
And scarcely greet me with that sun, thine eye
When love, converted from the thing it was,
Shall reasons find of settled gravity.*

Sonnets
WILLIAM SHAKESPEARE

Fashion is a form of ugliness so intolerable that we have to alter it every six months.

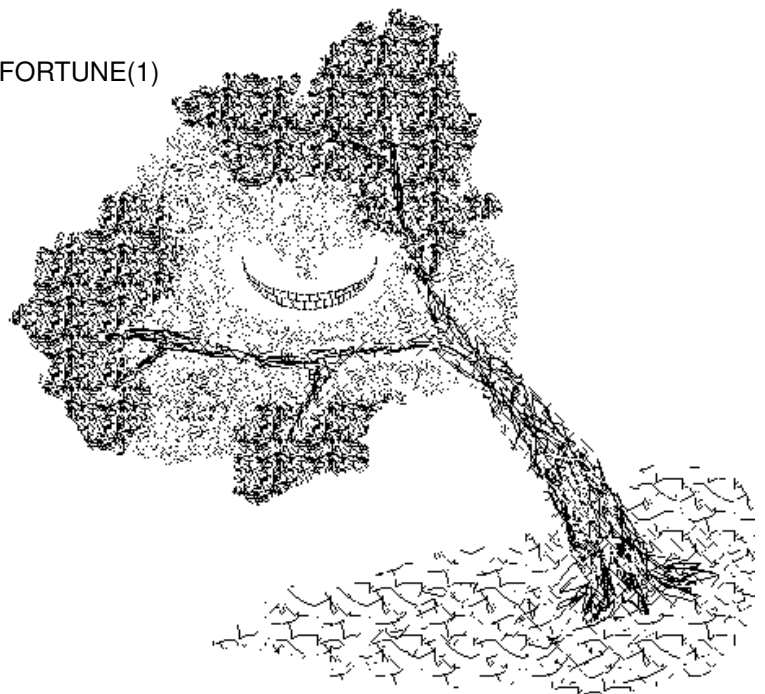
OSCAR WILDE

There is a certain relief in change, even though it be from bad to worse; as I have found in travelling in a stage-coach, that it is often a comfort to shift one's position and be bruised in a new place.

WASHINGTON IRVING

To determine how long it will take to write and debug a program, take your best estimate, multiply that by two, add one, and convert to the next higher units.

FORTUNE(1)



Reshaping a Database

The reshaping and restructuring of a *FirstBase* database is covered in this chapter. A slight understanding of *FirstBase* indexes is assumed.

The *FirstBase* tools covered in this chapter are *dbdcnv(1)* and *dbcgen(1)*.

Some other methods for doing database conversions are presented at the end of this chapter.

Converting a Database

FirstBase provides tools that allow any database to have new fields added and existing fields altered or removed from the database.

This process creates an entirely new database — and is called converting a database.

This database conversion allows molding of a new database without having to re-enter any existing data. It also provides a mechanism to add empty fields to a record.

The Conversion (or database) Generator generates a new database record for each indexed record in the current database. This allows a highly selective process of filtering and restructuring.

reminder

You will get exactly the same number of processed records as there are entries in your **index**! If this index is only a selected portion of your database, then that is all that will be processed.

The fields that are in each of these new records in the new database are controlled by the contents of the conversion dictionary.

This chapter discusses how to create and define a conversion dictionary, as well as how to actually generate a converted database.

The Conversion Dictionary

basic concepts

The conversion dictionary is much the same as a database dictionary in that it is a list of field names, each associated with a data type and a size or field length.

Unlike other generator dictionaries, the conversion dictionary can also contain fields that do not exist in the current, original database.

These new fields will be added to each new record but will not have a value in them. Fields in the conversion dictionary that do exist in the current database can be placed into the new database intact, or with their sizes altered.

Uses of Database Conversions

There are many uses for this database generator tool.

examples

Fields that were defined too short can have their length increased. For example, if we defined a ‘Last Name’ field as having a maximum of 25 characters, and then later, after many records have been created and we determine that 25 characters is not enough, we can use the conversion tools to lengthen the field to 40 characters.

Similarly, a field can be shortened during database conversion.

However, since *FirstBase* handles variable length records — meaning there is no extra storage for fields that do not take up their entire maximum length — shortening a field is rarely done. But it will work.

UNIX Note

For UNIX people: if the only thing you need to do is change the sizes of fields, you may be able to change the dictionary directly.

It is allowable to use a UNIX editor on the current database dictionary to expand or contract the field size.

changing field sizes

Be careful, though.

Bumping the size is never a problem due to the way the *FirstBase* database engine works. However, decreasing the size should only be done if the largest string length for that field is known.

In any case, *dbcheck(8)* should be used to make sure you have not damaged any definitions. Use the **-clp** flags, also.

changing field types

Additionally, you must use a UNIX editor if you want to change the type of a database field.

These changes, too, can be very delicate.

Knowledge of the data being stored is needed, as well as an understanding of data types in general.

For example, converting a numeric (**n**) field to an alpha (**a**) field will always work, but the converse may not.

more
examples

As another example, if after some use of a database you determine that new fields are also desired in the database — say, the tracking of an office phone number — you can add this field via database conversion.

Of course, the field will be empty since it previously did not exist, but the field will be there just as if you had defined it from the start.

A third example involves taking a very selective index and generating a new database using this index. As mentioned, only these indexed records will exist in the new database.

If you use the same structure of database for the conversion as was originally defined, then the new database will be compatible with the old or current database.

Databases that are compatible — or that have the exact same data definition — can be concatenated together using still another *FirstBase* tool. This tool, *dbcats(1)*, is discussed in Chapter 13, "Database Concatenation"

hints

This last sample provides an example of physically filtering out unwanted records from a database — perhaps for storing the database permanently off line.

However, if you are merely trying to impose some other indexed structure or want to reorder the fields of an existing database, these tasks can be done without the conversion process.

See Chapter 6, "Creating Custom Screens and Views" and Chapter 8, "Creating an Index" for more details.

Getting Started

The first thing to do is to decide exactly how you want to structure or organize the new database.

The order of the fields can be changed for display and output at any time easily, so do not worry about what order the fields are listed.

The next step is to define the conversion. A Define Conversion tool is provided. This process will create and edit a conversion dictionary.

As expected, the conversion dictionary editing is done in the same way as most other dictionary editors.

Creating the Conversion Definition

name
database
and index

The first thing you want to do is to name your database and index. This can be done by using the environment command screen described in "Changing a File Name" on page 3-5.

For our example here, we will add a new field to the database and re-define an existing fields size — all in one pass — using the *FirstBase* database conversion tools.

From the *FirstBase* MAIN menu choose the selection titled **Define Conversion** to start the *FirstBase* tool for defining a database conversion.

The Define Conversion Screen

The following screen fragment shows what your screen will look like when you start the Define Conversion tool:

```

FirstBase 6.5: dbdcnv          FirstBase Software          Status: Define Conversion
  Field          Type          Size          Conversion Status
  ----          -

```

The idea is to build a list of database fields that the conversion generator will use to create the new database.

Again, the fields do not need to exist in the current database dictionary.

The 'Conversion Status' column will display the status of each field.

The Field Column

@

To begin entering conversion fields, use an '@' to go into 'auto add' mode. In this mode, you will continually enter fields until the standard END keystroke is used from the field column.

The Define Conversion tool will attempt to put something reasonable into each column if you hit a <RETURN> without entering any characters.

This feature can be used to quickly get an image of the database dictionary into the conversion dictionary.

<RETURN>

For example, after going into add mode, hit a <RETURN>. The first field of the database will be inserted for you. In our example, this field is named 'Last Name'.

Note that unique naming of fields is enforced here. That is, it is an error to try and enter a field name that is already in use in the conversion dictionary.

The Type Column

After entering a field name in the field column, the cursor will move to the **Type** column.

But, if the conversion field you are working already exists and is not a new field, the Define Conversion tool will place the current type of the field in this column, and move on to the **Size** column.

However, if you are entering a new field, the cursor will stop at this **Type** column to allow you to enter the type of the new database field.

The default **Type** is for alphanumeric data.

The Size Column

The **size** column is used to place the size you want the field to have in the new database.

Again, a default here will store the same value there that is currently being used by the database.

The size indication is used to determine if the conversion field will be an 'expanded', 'image', or 'truncated' field.

But, if the field is a new field that does not exist in the current database, then the word 'new' will appear in the **Conversion Status** column. The default field size is 10.

The Conversion Status

No entry is required for this column — it is displayed to help you visualize what the effects of the conversion will be and to determine whether you are really doing what you want to with the conversion dictionary.

Miscellaneous Conversion Information

The Define Conversion tool also provides a method of changing the default name of the new database. The default name is 'convrt'.

m

You can get to this miscellaneous screen using the **m** command from the Any Change screen. Also, the Define Conversion tool will drop into this miscellaneous screen when you END out of the program.

Making Corrections

-

To exit add mode, use the standard END keystroke '-' from the field column. This keystroke will put the cursor back onto the command line at the bottom of the screen, where corrections can be made to the conversion dictionary.

Like all other *FirstBase* tools, you can make corrections to any of the fields before you exit from the *FirstBase* tool. This is accomplished

making
corrections

by typing the number of the line you want to change at the Any Change screen.

Once the cursor is on the line you want to change, a <CONTROL>-X will move you across columns without disturbing the data that is there.

When the cursor is on the column you want to change, make your modifications, hit a <RETURN>, and continue.

The following is a sample fragment of a completed conversion dictionary screen:

Field	Type	Size	Conversion Status
1) Last Name	a	35	<image>
2) First Name	a	25	expanded
3) Initial	a	1	<image>
4) Telephone	a	15	<image>
5) Work Phone	a	15	new

—

Use the END keystroke to exit this screen

As you exit, the Define Conversion tool will place you in the miscellaneous screen.

When done there, use the END keystroke again to exit.

Generating a Converted Database

Now comes the easy part — generating the conversion.

If you have defined the conversion or gone into the Define Conversion program, then when you exit you will be asked whether you want to run the conversion generator.

y

If you respond with a y, the Conversion Generator will be run for the labels you have just defined.

Or you can say go back to the menus, locate the Conversion Generator tool in the MAIN menu.

No matter how you start the Conversion Generator, you will get the same screens and results — assuming that the same database and index names are used.

Automatic Index Regeneration

The first thing that will happen (if needed) is the index will be regenerated. During this phase, the status area on the header line will say 'Auto Regeneration' and the middle of the screen will say 'Working'.

When the index is regenerated, the generator will move on to the next screen automatically.

Interactive Generation

generation

When you are interactively generating a conversion, the generator will display a listing of the conversion along with the miscellaneous information.

This screen represents the generator's idea or 'picture' of what it thinks you want to generate. At this point, the generator will pause, waiting for an answer.

If for some reason you do not want to continue, answer the 'If accurate' prompt with a <RETURN>.

y

Once you respond **y** to the 'If accurate' prompt, the Conversion Generator will fly through the database, its index and the conversion dictionary you have defined, creating a new database for you.

A counter appearing in the lower right hand quadrant of the terminal will display the number of records processed.

For each indexed record, the conversion generator will create a single record in the new database.

The end result will be a new *FirstBase* database, and dictionary, as well as all the needed support files (a database dictionary and a map), all tied to the output name as defined in the conversion dictionary.

This new database is automatically 'cleaned' after being generated. See *dbclean(1)* for more details.

The newly generated database can immediately be used by any other *FirstBase* tool. Remember to regenerate any needed indexes, also.

Note that the newly converted database dictionary will *not* contain references to autoindexes that the original database contained. This is a safeguard feature — not a bug.

To add an autoindex to the new database, use the methods from "AutoIndex the Database" on page 4-8

To remove an index, but leave the index dictionary, use *rmdir(8)*.

Database Checking

Once you have generated a database, it is good practice to use the database check facility, *dbcheck(8)*, to make sure the newly created database is healthy. If you can, use the **-clp** flags to ensure database integrity.

Other Database Conversion Methods

There are other ways to convert existing databases new databases.

First off, you could use *dbemit(1)* to create a *loadfile*, modify the data dictionary, and *dbload(1)* the *loadfile* back into place. Remember to use the **-q** flag if you have embedded quotes, or change the separator, and *mvdb(8)* or *rmdb(8)* to move or remove the old database before reloading.

See Chapter 16, "Downloading and Uploading Databases".

Another method of converting a database would be to use the *dbsql(1)* tool to create a new databases from an SQL selection.

See the manual pages on these tools for more details.

And, perhaps a conversion is not even needed. You can always mask out extra fields using a screen or view dictionary. See Chapter 6, "Creating Custom Screens and Views".

Also, if you define the original database with excess fields, you can always go back and merely add these fields to a view dictionary without having to do a complete conversion.

Additionally, see the UNIX Note on page page 12-2.

Note: DO NOT use a UNIX editor to insert or delete fields from a database dictionary that has live data in the database file.

You must use *dbcgen(1)* to do this, or one of the above methods.

Summary

This chapter has discussed some of the aspects of generating a converted database using the Conversion Dictionary Editor and the Conversion Generator.

Conversions allow the *FirstBase* user to easily restructure a database at any time without re-entering existing data.

For more details see the manual pages on *dbdcnv(1)* and *dbcgen(1)*.

A countryman between two lawyers is like a fish between two cats.

BENJAMIN FRANKLIN

If all the girls attending it [the Yale Prom] were laid end to end, I wouldn't be at all surprised.

DOROTHY PARKER

More ways of killing a cat than choking her with cream.

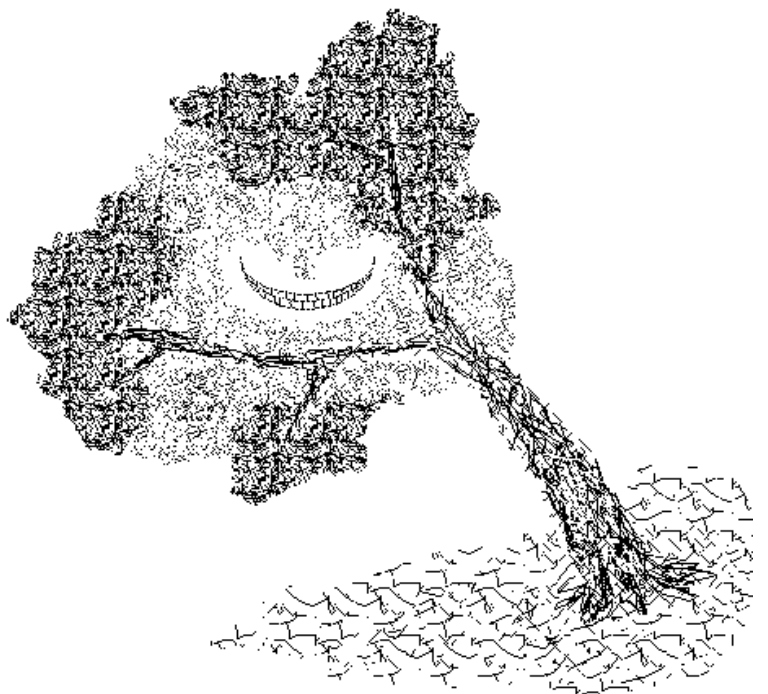
CHARLES KINGSLEY

It all comes to the same thing at the end.

Any Wife to Any Husband
ROBERT BROWNING

I could play Eracles rarely, or a part to tear a cat in, to make all split.

A Midsummer Night's Dream
WILLIAM SHAKESPEARE



Database Concatenation

This chapter covers database concatenation, the connecting together of two or more like databases. Indexes are not used for this *FirstBase* tool.

The *FirstBase* tool covered here is *dbcat(1)*.

Connecting Databases

Database concatenation means connecting two or more databases, end to end, to create a larger database.

The resulting database contains all the records of each of the separate databases fed to the database concatenation tool.

No indexes are used — all non-deleted records from each separate database will be written to the new database.

The Database Concatenation tool, *dbcat(1)*, is not a generator — it does not require a generator dictionary.

The only items required by this tool are two or more compatible databases.

basic
concepts

Compatible databases have **exactly** the same shape.

Usually, like databases will have the same dictionary, although this is not a necessity.

Again, the databases you concatenate together must have the same fields in the same order, with all of the corresponding fields being the same **type** and **size**.

This is what is meant by ‘compatible databases’.

For example, you might want to have two or more people keep their own databases in their areas or machines until the end of the month.

Then all the databases can be collected together and concatenated so as to produce a single report generated from all of the data. Or to take the whole database off line.

Running The Concatenation Tool

From the *FirstBase* system menus, you will only be able to concatenate two databases together.

If you use the Concatenation Tool from a UNIX shell, such as *cs*(1) or *sh*(1), you can concatenate as many databases together as you want.

In this case, all the names of the separate databases will come from the UNIX command line.

name
database

Since our examples in this manual are being done from the *FirstBase* menus, we will describe the actions of *dbc*(1) as done from these menus.

From the menus, you will need to make sure the first database is named properly. See "Changing a File Name" on page 3-5.

concat

The database concatenation tool is located in the *FirstBase* TOOLS menu, and is labeled **concat**.

enter
input and
output
database
names

Once selected, the Concatenation Tool (*dbc*(1)) will use the default or named database as its first database to concatenate.

Then you will be prompted for the name of the second database, as well as the name of an output database. The default for the output database name is 'concat'.

Once all the file names have been entered, *dbc*(1) will pause to make sure that you want to continue. You must respond with a **y** to continue the concatenation process.

y

Once started, *dbc*(1) will process each database, placing all undeleted records into the output database.

The footer line at the bottom of the screen will indicate the name of each database as it is being processed.

Upon completion of the Database Concatenation tool, the resulting database will be cleaned automatically. See *dbclean*(1) for more.

All support files (the database dictionary and the database map) will be produced by *dbc*(1) as well.

The output database can immediately be used by any other *FirstBase* tools.

Summary

This chapter has discussed the more simple aspects of database concatenation.

There is not much to this tool — it will do as you would expect as long as you are careful to feed it databases that have the same structure.

If you want more details, see the manual page on *dbc*(1).

Perhaps my name too will be linked with theirs.

OVID
43 B.C. — A.D. 17

*Woe unto them that join house to house,
that lay field to field, till there be no place.*

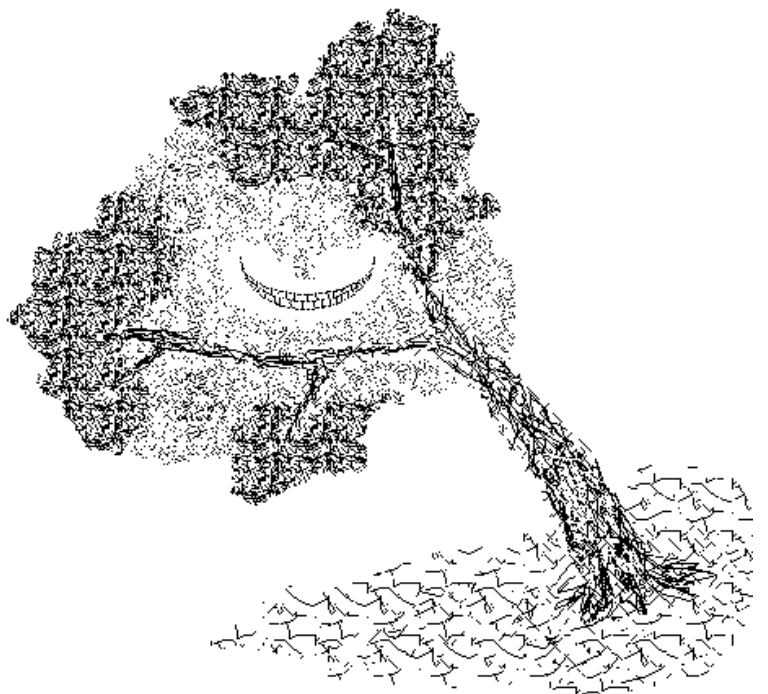
Isaiah 5:8
BIBLE

*Three poets, in three distant ages born,
Greece, Italy, and England did adorn.
The first in loftiness, of thoughts surpass'd;
The next in majesty, in both the last;
The force of nature could no farther go;
To make a third she join'd the former two.*

JOHN DRYDEN

Modern man is the link between apes and human beings.

FORTUNE(1)



Database Joining

This chapter covers database joining, the placing of two databases side by side into a single database.

Indexes are used for each database to be joined. It is assumed that you understand *FirstBase* indexes.

The *FirstBase* tool covered here is *dbjoin(1)*.

FirstBase also supports dynamic joins — a feature that allows the use of joined records, without the physical join.

Furthermore, the SQL tool, *dbsql(1)*, provides a join mechanism via the `create` and `select` statements.

The SQL join is a many-to-many join, while the *dbjoin(1)* and the dynamic join mechanisms are many-to-one or one-to-one.

For more details, read about *Linked Fields* in the manual pages for *dbdbas(1)* and *dbedit(1)*, as well as these other manual pages.

Joining Two Databases

Database joining means that two databases that are usually unlike in most aspects are placed side by side into a single output database.

Each record of the resulting database is made up of two records, one from each of the input databases. This produces a longer record that can then be accessed and used as a single unit or record.

The Database Joining tool, *dbjoin(1)*, is not a generator — it does not require a generator dictionary.

The only thing required by this tool is two indexed databases.

Each database should have a common field that exists in both databases — the resulting database is joined using this common field.

Database joining revolves around the common field between the databases. This common field is indexed for each database.

The indexes are then processed during database joining.

Records of each database are joined together when the values of this common field match exactly.

basic
concepts

example

A good example of database joining would involve student records.

Say that two databases are being kept separate — one containing student grades for a single semester, the other containing static student information such as parents name and home address.

The common field in this student example would be the student matriculation number — the unique identification number assigned to each student.

After joining on this common field, the result is a database that contains exactly one record for each student matric number — but has all the fields from both separate databases.

Getting Started

First, each database is indexed by the common field. This common field can be named the same, or differently, in each database

In our student example, this field is the Matric number.

At this point, we have two databases and two indexes.

Each of the other fields in these databases, other than the common field we are joining on, needs to have unique names.

Remember: you cannot have a *FirstBase* database that has two fields called 'Last Name' in it.

If you join two databases together that each have a field called 'Last Name' (assuming this is NOT the common field), the resulting database would have two fields named 'Last Name', and this is not allowed.

If this field naming is a problem, i.e. if you have many common fields among your databases, you can do one of few things.

Either change the names of these duplicate field names in one of the databases, use the database conversion program (`dbcgen(1)`) to filter out the fields that are duplicated, or join using `dbsql(1)`.

If you are going to use `dbjoin(1)`, leave one common field to join each record with.

Running The Database Join Tool

From the menus, you will need to make sure the first database and index are named properly. This can be done by using the environment command screen described in "Changing a File Name" on page 3-5.

The Database Join tool is located in the *FirstBase* TOOLS menu, and is labeled **join**.

reminder

You will get exactly the same number of processed records as there are entries in your **indexes!** If these indexes represent only a selected portion of your databases, then that is all that will be joined.

enter
secondary
and
output
database
names

Once selected, the Joining Tool (*dbjoin(1)*) will use the default or named database as its first database to join.

Then you will be prompted for the name of the second database, as well as the name of an output database. The default database for the output database is 'join'.

Automatic Index Regeneration

The first thing that will happen (if needed) is the indexes will be regenerated. During this phase, the status area on the header line will say 'Auto Regeneration' and the middle of the screen will say 'Working'.

After the indexes are regenerated, the Database Join tool will move on to the next screen automatically.

Interactive Generation

When you are interactively doing a database join, the tool will display a list of the field names of each database, with the common field listed at the top of the screen.

generation

This screen represents the tool's idea or 'picture' of what it thinks you want to produce. At this point, the tool will pause, waiting for an answer.

If for some reason you do not want to continue, answer the 'If accurate' prompt with a <RETURN>.

y

Once you respond **y** to the 'If accurate' prompt, the join tool will fly through the two databases, and their indexes, creating a new database for you.

A counter appearing in the lower right hand quadrant of the terminal will display the number of records processed.

For each pair of indexed records, the Database Join tool will create a single record in the new database.

The end result will be a new *FirstBase* database, as well as all the needed support files (a database dictionary and a map), all tied to the output database name provided.

This new database is automatically ‘cleaned’ after being generated. See *dbclean(1)* for more details.

The newly generated database can immediately be used by any other *FirstBase* tool.

It is possible to force *dbjoin(1)* to create an output record even for those records that do not share a common field value.

In this case, when the record is written to the new database, the extra fields from the ‘other’ database are empty.

Summary

This chapter has discussed simple aspects of database joining. There is not much to this tool — it will do as you would expect.

If you want more details, see the manual page on *dbjoin(1)*.

Times change, and we change with them.

EMPEROR LOTHAR I

*Let me not to the marriage of true minds
Admit impediments. Love is not love
Which alters when it alteration finds,
Or bends with the remover to remove:
O, no! it is an ever fixed mark,
That looks on tempests and is never shaken...*

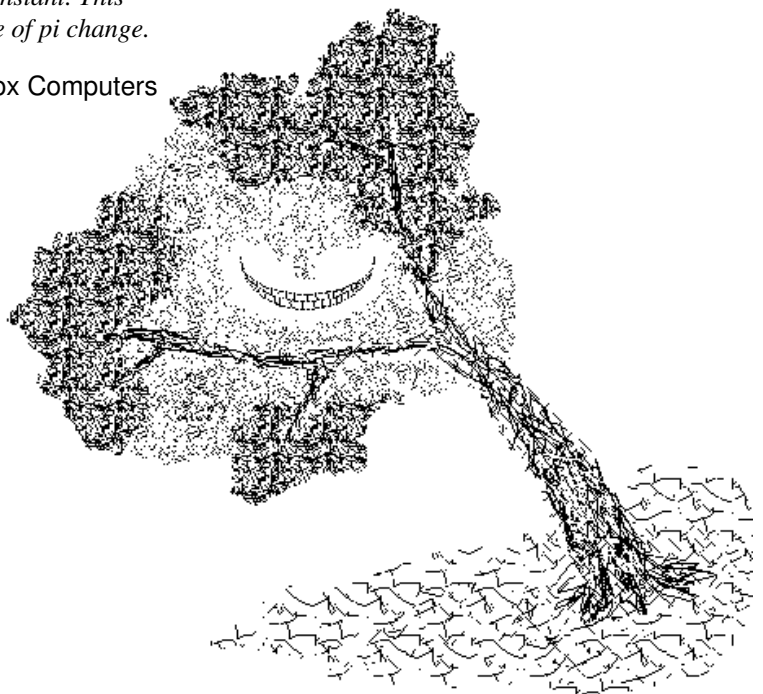
Sonnets
WILLIAM SHAKESPEARE

The more things change, the more they are the same.

ALPHONSE KARR

The primary purpose of the DATA statement is to give names to constants; instead of referring to pi as 3.141592653589793 at every appearance, the variable PI can be given that value with a DATA statement and used instead of the longer form of the constant. This also simplifies modifying the program, should the value of pi change.

FORTTRAN manual for Xerox Computers



Global Database Updates

This chapter details the updating of fields across many records in a *FirstBase* database. A slight understanding of *FirstBase* indexes is assumed.

The *FirstBase* tools used are *dbdupd(1)* and *dbugen(1)*.

Another *FirstBase* tool, *dbrload(1)*, provides a replacement/loading mechanism that may fit an applications update needs more precisely. Additionally, *dbmacro(1)* can also be used for some of these situations. See these manual pages for more details.

Updating a Database

FirstBase has tools that update or modify particular fields of every indexed record in a database.

The fields are modified according to values and formulas stored in the update dictionary.

reminder

You will get exactly the same number of processed records as there are entries in your **index**! If this index is only a selected portion of your database, then that is all that will be processed.

A special mechanism is included that will do global deletions. See the section on update strings below for details.

The Update Dictionary

basic concepts

The update dictionary is merely a list of database field names with associated update values.

These update values are used as the new field values for every indexed record when the update generator is used.

For numeric database fields, the update value is stated as a formula, and can reference other numeric fields.

Uses of Database Updates

There are many uses for this database generator tool, especially used in conjunction with a very selective *FirstBase* index.

As a simple example, if you have a database that contains client

examples

records, each with a 'Balance Due' field, and you need to invoice each client (or each *indexed* client!) a \$200 fee, these *FirstBase* tools can be used to update each field of these clients records. The update would be phrased as a formula which would add the \$200 to the current value.

As another example, if you have employee records that store each employees hours and pay rate, the update tools could be used to formulate the pay due and to physically store this value in a field.

This updated field could be moved to a new database or used as is.

Using this update method is different than using a Formula field, since Formula fields are virtual fields, and do not really exist.

Getting Started

The first step is to define the update using the Define Update tool.

This process will create and edit a update dictionary.

As expected, the update dictionary editing is done in the same way as the other *FirstBase* dictionary editors.

Creating the Update Definition

name
database
and index

The first thing you want to do is to name your database and index. This can be done by using the environment command screen described in "Changing a File Name" on page 3-5.

For our example here, we will add a constant value to a field in the database — using the *FirstBase* Update Generator.

From the *FirstBase* MAIN menu choose the selection titled **Define Update** to start the *FirstBase* tool for defining a database update.

The Define Update Screen

The following screen fragment shows what your screen will look like when you start the Define Update tool:

FirstBase 6.5: dbdupd		FirstBase Software		Status: Define Updates
Field	Type	Size	Update String	
----	----	----	-----	

The idea is to build a list of update fields and values that the Update Generator will use to globally update the database.

The **Type** and **Size** columns are displayed to help you use proper update values

The Field Column

@

To begin entering update fields, use an '@' to go into 'auto add' mode. In this mode, you will continually choose fields and update string values until the standard END keystroke is used from the field column choice screen.

choose
update
field

To fill in the Field column, you will be given a standard Chose Field screen, as described in "FirstBase Choose Field Mechanism" on page 2-12.

Select the field you want to update, and hit a <RETURN. The screen will then be redrawn with the Type and Size of the field displayed.

The Update String Column

enter
update
string

After selecting a field name in the field column, the cursor will move to the Update String column. Here you enter an update value for the field you named in the field column.

For alpha, alphanumeric, and date fields, the value you enter is stored as is into the database records that are processed.

When updating date fields, make sure you enter exactly 6 numbers, no slashes. For example, '013192' for January 31, 1992.

tips

For numeric type fields (numeric, Numeric, dollar, and float) the update string is processed as if it was a 'formula'.

This formula is used the same way as Formula fields are defined, as described in the manual page for *dbdbas(1)*.

Formulas use numbers to represent fields (e.g. 16 for field 16) and a prepended 'C' or 'c' for constants (e.g. C4.5 for 4.5).

Addition, subtraction, multiplication and division are the operators.

References to other formulas will not work.

All formulas read from left to right, and there is no precedence among the operators.

Do not use blanks in your formula specification.

The precision of a formula is specified by appending a colon (:) to the formula followed by a number. Default precision is two.

hints

Again, particular attention must be paid to the update string if the field to be updated is numeric in nature.

Specifically, if the precision of constants and other numbers are not set properly, the results will not be as expected.

another
example

For example, to update a dollar field to contain \$123.45, the actual update string required is “C12345:0”, since *FirstBase* ‘knows’ where the decimal goes for all dollar fields.

deleting
records

Deleting Records

A special meta-update string, **\$DELETE** is used to delete all records that are touched during running of the Update Generator.

In this case, the actual contents of the field are not modified.

Again, use the string **\$DELETE** as an update string for any field to delete all of the records touched during the update process.

—

Making Corrections

To exit add mode, use the standard END keystroke ‘-’ from the standard Choose Field screen.

making
corrections

This keystroke will put the cursor back onto the command line at the bottom of the screen, where corrections can be made to the update dictionary.

Like all other *FirstBase* tools, you can make corrections to any of the fields before you exit from the *FirstBase* tool.

This is accomplished by typing the number of the line you want to change at the Any Change screen.

—

Once the cursor is on the line you want to change, a <CONTROL>-X will move you across columns without disturbing the data that is there.

Use the END keystroke to exit the in the Define Update program.

The following fragment shows a simple update dictionary entry for the Balance Due example:

FirstBase 6.5: dbdupd		FirstBase Software		Status: Define Updates
Field	Type	Size	Update String	
----	----	----	-----	
1) Balance	\$	15	2+C20000:0	

The ‘2’ in the update string refers to the field numbered 2 in our database.

This is the Balance field itself, making the whole formula **2+C20000:0** actually add \$200 to the Balance field (when the generator is run).

Generating Database Updates

Now comes the easy part — generating the update.

If you have defined the update or gone into the Define Update program, then when you exit you will be asked whether you want to run the update generator.

y

If you respond with a y, the Update Generator will be run for the labels you have just defined.

Or you can say go back to the menus, locate the Update Generator tool in the MAIN menu.

No matter how you start the Update Generator, you will get the same screens and results — assuming that the same database and index names are used.

Automatic Index Regeneration

The first thing that will happen (if needed) is the index will be regenerated. During this phase, the status area on the header line will say ‘Auto Regeneration’ and the middle of the screen will say ‘Working’.

When the index is regenerated, the generator will move on to the next screen automatically.

Interactive Generation

When you are interactively generating an update, the generator will display a listing of the update fields and update strings.

This screen represents the generator's idea or ‘picture’ of what it thinks you want to generate. At this point, the generator will pause, waiting for an answer, and display the ‘If accurate’ prompt.

If for some reason you do not want to continue, use a <RETURN>.

y

If you respond y then for each indexed record, the update generator will update that record using the defined update dictionary.

A counter in the will display the number of records processed.

Summary

This chapter has discussed some of the aspects of generating global and selective updates to a *FirstBase* database using the update dictionary editor and the update generator.

Updates allow the *FirstBase* user to plug values and formulas into selected fields of database records.

For more details see the manual pages on *dbdupd(1)* and *dbugen(1)*.

*All things are full of labour; man cannot utter it:
the eye is not satisfied with seeing,
nor the ear filled with hearing.
The thing that hath been, it is that which shall be;
and that which is done is that which shall be done:
and there is no new thing under the sun.*

Ecclesiastes 1:8
BIBLE

*You, I am sure, will forgive me for sincerely remarking that you might
curb your magnanimity, and be more of an artist, and load every rift
of your subject with ore.*

JOHN KEATS

Sure the poet...spewed up a good lump of clotted nonsense at once.

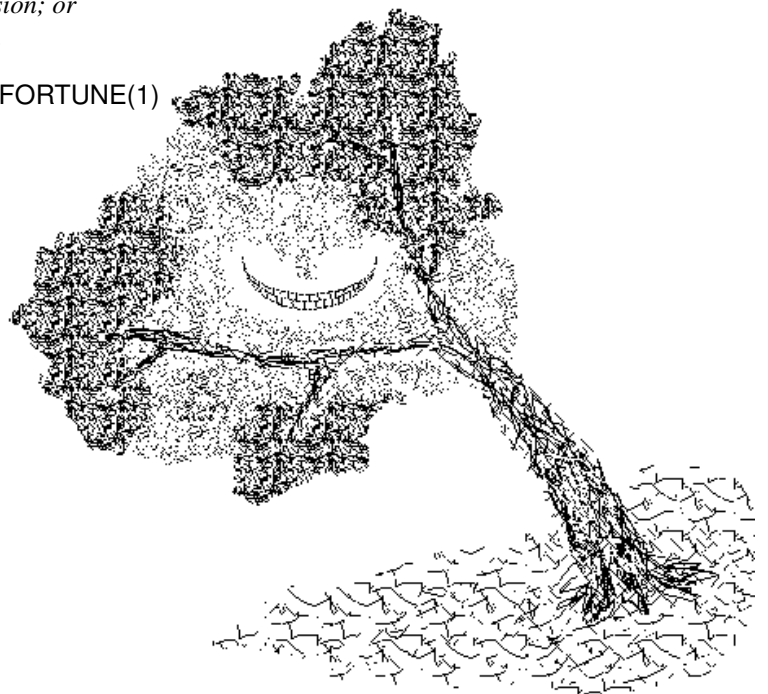
JOHN DRYDEN

*The bookful blockhead, ignorantly read,
With loads of learned lumber in his head.*

ALEXANDER POPE

*There are three possibilities: Pioneer's solar panel has turned away
from the sun; there's a large meteor blocking transmission; or
someone loaded Star Trek 3.2 into our video processor.*

FORTUNE(1)



Downloading and Uploading Databases

This chapter covers the *FirstBase* tools that provide database downloading or database uploading. These tools are *dbemit(1)* and *dbload(1)*.

Emitting Database Values

FirstBase provides a tool that allows the emission of database values. These values are produced as token separated values, with one record per line. The default token is a comma.

If desired, alphanumeric fields can be enclosed in quotation marks.

emit

This very simple tool emits fields of non-deleted records from a database into an ascii file. The format of the emitted data is compatible with many other database and statistic packages.

A standard *FirstBase* screen dictionary can be used to mask and alter the order of the database fields. Also, an optional index can be used.

The emitting tool, *dbemit(1)*, is located in the *FirstBase* TOOLS menu, and is labeled **emit**.

When used from the menus, you will be prompted for the name of an output file. *Dbemit(1)* adds a '.emit' extension to the filename entered.

Dbemit can also emit its data to standard output.

See the manual page on *dbemit(1)* for more details.

Uploading into *FirstBase*

**define
database**

FirstBase also provides a method of uploading data into the a *FirstBase* database.

To begin, you will need to define a database. See Chapter 4, "Defining a FirstBase Database" for details.

The tool for uploading data, *dbload(1)*, is located in the *FirstBase* TOOLS menu, and is labeled **load**.

This tool will prompt for a load file name. The file name is left intact — that is, no extensions are added to it by the loading tool.

load

The data in the load file is expected to be one record per line, with a comma separating each field value.

Alpha fields do not need to be quoted, unless the field contains embedded commas.

Furthermore, the standard UNIX method of escaping characters using the backslash (\) is supported.

At the completion of the loading phase, *dbload(1)* will automatically run the database cleaning program, *dbclean(1)*.

All support files are generated by *dbload(1)* — the database file and map — making the resulting database immediately usable by any tool in *FirstBase*.

It is a good idea to run the *dbcheck(8)* facility across any database that has just been loaded before the database is used.

This integrity check will help find any errors with the load file.

Be sure to use the **-clp** flags during the check.

Summary

This chapter has touched on the *FirstBase* tools that allow database downloading and uploading.

There is not much to these tools — they do pretty much as expected.

For a bit more information, see the manual pages on *dbemit(1)* and *dbload(1)*.

Another *FirstBase* tool, *dbrload(1)*, provides a replacement/loading mechanism that may fit an applications update needs more precisely.

*What's past, and what's to come is strew'd with husks
And formless ruin of oblivion.*

Troilus and Cressida
WILLIAM SHAKESPEARE

*I have seen
A curious child, who dwelt upon a tract
Of inland ground applying to his ear
The convolutions of a smooth-lipped shell;
To which, in silence hushed, his very soul
Listened intently; and his countenance soon
Brightened with joy; for from within were heard
Murmurings, whereby the monitor expressed
Mysterious union with its native sea.*

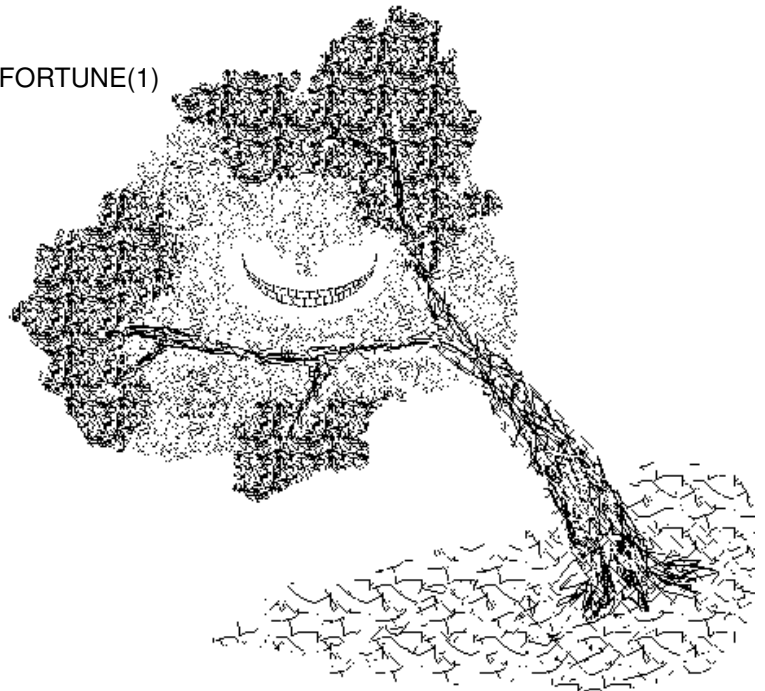
The Excursion
WILLIAM WORDSWORTH

*The humblest citizen of all the land, when clad in the armor of a
righteous cause, is stronger than all the hosts of error.*

WILLIAM JENNINGS BRYAN

*The first bug to hit a clean windshield lands directly
in front of your eyes.*

FORTUNE(1)



Creating *FirstBase* Menus

This chapter outlines the components of menu dictionaries used by the *FirstBase* menu tool, *dbshell*(1), to produce menus. The menu tool blends the power of UNIX shell scripts with the convenience of menus.

Menu Dictionaries

Like other *FirstBase* tools, *dbshell* is dictionary driven. The menu dictionary is a free form text file created using the text editor of your choice, like *vi*(1).

basic
concepts

This file is used by *dbshell* to display a screen and provide commands for requested menu selections.

The *dbshell* dictionary consists of two main parts, the *screen* section and the *actions* section.

The *screen* section is the portion of the menu dictionary that will be displayed during the *dbshell*(1) session. The *actions* section is the area where shell commands or scripts are stored.

The Screen Section

screen

When *dbshell* is invoked, the display on the monitor will appear almost exactly as listed in the *screen* section of the menu dictionary.

\$token

Words and tokens can be displayed in reverse video by prepending them with a dollar sign (\$). Use an underscore (_) to force reverse video of a blank.

%

The *screen* section of the menu dictionary, a maximum of 21 lines long, will be displayed between the standard *FirstBase* header and footer lines.

A line with a single percent sign (%) in the menu dictionary marks the end of this displayable section.

The Actions Section

actions

The *actions* section of the menu dictionary, beginning after the percent marker, consists of one or more labeled sections of text. Each section is a different action.

\$label

Each action consists of a dollar sign (\$) and a user defined label, followed by one or more lines that comprise the action.

The label represents the characters that will need to be typed to invoke the corresponding action. Only enough characters to distinctly identify the selection need be typed.

As many lines as needed can be used for each actions sub script.

The end of an individual action script is the beginning of the next action script, as denoted by the new label, or the end of the menu dictionary file.

Example

Here is a very small sample *dbshell(1)* dictionary/text file:

```
    $Major_Functions      $Minor_Functions
$1 - Choice One          $mail - Send Mail
$2 - Edit Database      $ls   - List Directory

                        $Sub_Menus

                        $print - Printout Sub Menu
                        $quit  - Quit

%

$1
    echo choice one

$2
    cd /usr/databases/work
    dbedit -d master -i coname -s invoice

$mail
    echo -n "Send mail to who? "
    read MAILTO
    echo "Enter mail message now."
    echo "End with a `.'`."
    mail $MAILTO

$ls -1
    ls -F

$print *
    PRINT

$q $EXIT 0
    echo Good Bye
```


Using `dbshell`, this dictionary sample would paint your display like:

```

FirstBase 6.5: dbshell      FirstBase Software      Status: SAMPLE

      Major Functions      Minor Functions

1 - Choice One           mail - Send Mail
2 - Edit Database       ls  - List Directory

      Sub Menus

print - Printout Menu
work  - Work Menu
quit  - Quit

```

Special Action Flags

There are three special flags that may follow the action label:

- 1 after executing this sub script, do NOT pause
- * denotes a sub menu dictionary
- \$EXIT causes a forced exit from *dbshell*

The sub menu marker (*) can have an optional directory following it. This directory will become the working directory during execution of the sub menu.

The \$EXIT flag can also list an argument that will be the exit value. For example, '\$EXIT 31' would cause a termination from the menu tool with the value 31.

The *screen* and *actions* portions of the menu dictionary are independent. The screen is usually set up with some sort of a title and comments that describe the given actions, but this is not a require-

Summary

This chapter has discussed the creation of *FirstBase* menu dictionaries. The menu tool, *dbshell* is a very powerful *FirstBase* tool that can be used to group commands and provide a framework for systems of commands.

See "Application Menus" on page 19-1 and *dbshell(1)* for more information.

Man is a tool making man.

BENJAMIN FRANKLIN

As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs.

MAURICE WILKES
discovers debugging, 1949

Every program has at least one bug and can be shortened by at least one instruction — from which, by induction, one can deduce that every program can be reduced to one instruction which doesn't work.

FORTUNE(1)

*We spend our lives in learning pilotage,
And grow good steersmen when the vessel's crank.*

The Wisdom of Eld
GEORGE MEREDITH

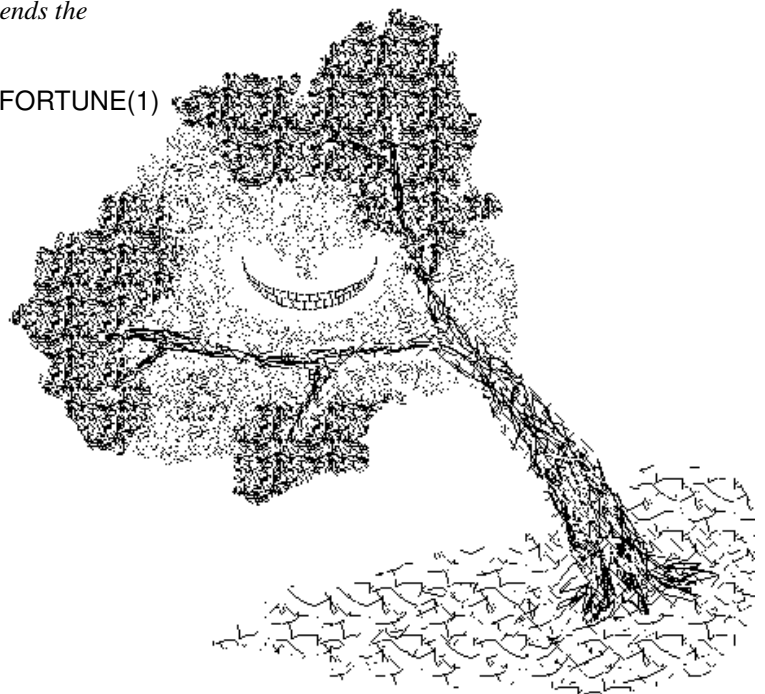
Education makes people easy to lead, but difficult to drive; easy to govern, but impossible to enslave.

LORD BROUGHAM

Arnold's Law of Documentation:

- (1) If it should exist, it doesn't.*
- (2) If it does exist, it's out of date.*
- (3) Only documentation for useless programs transcends the first two laws*

FORTUNE(1)



This chapter covers tasks that are managerial in nature. These tasks include pointers on database security, checking database integrity, and *FirstBase* environment control hints.

Database Security

Most of the database security is handled in the standard UNIX method of file protection.

UNIX provides three levels of protection, user, group, and other. It is beyond the scope of this manual to discuss these mechanisms.

Suffice it to say that *FirstBase* will automatically conform to these UNIX protection levels.

FirstBase also provides another very simple method of database security — the database password.

Each *FirstBase* database can have a password associated with it. This password will be prompted for whenever the database editor is invoked on the database.

password

This database password only protects from the use of the database editor on the database in question. All other tools will still work, meaning a user can see what data is there, but cannot change it with the editor.

scanner

There are forms of the database editors known as scanners. These tools, named *dbscan(1)* and *dbvscan(1)* behave the same as the database editor, but they do not allow record modifications.

The scanners can be used when a database is password protected.

Again, the file protection mechanisms of UNIX are the first levels of protection for the *FirstBase*. Use the database password for an additional level if needed.

To define or change a database password, use the *FirstBase* tool *dbpasswd(1)*. See the manual page on this tool for more details.

Multi User Databases

record
locking

FirstBase automatically takes care of record locking using UNIX system calls. The main use for this feature is to allow two or more people editing a single database at one time. For the most part, the outlying

generators (*dbpgen(1)*, etc.) do not need to pay attention to locked records.

However, the `LOCKLEVEL` environment variable can be used to force even generators to avoid locked records. See *setup(5)* for more.

Database Integrity

FirstBase provides a tool that will check the integrity of a *FirstBase* database. This tool, *dbcheck(8)*, is analogous to the UNIX tool *fsck(8)* used for checking the integrity of a UNIX file system.

dbcheck

Dbcheck, the database integrity checker, will print statistics and details about *FirstBase* databases.

These details include the record count, the deleted record count, and the percentage of bytes in use in the database.

To invoke with the most stringent integrity tests, use the `-clp` flags. For more details, see the manual page on *dbcheck(8)*.

Database Cleaning

dbclean

Since the *FirstBase* system uses variable length records, databases can become frayed and fragmented, using up more physical space than they really need to.

FirstBase provides a tool that will recreate the database, cleaning out all fragmented bytes and any free space in the database file.

This cleaning program, *dbclean(1)*, will also produce a database map, a necessary support file to map out the records so other programs know where to find these database records.

Note: *dbclean(1)* must be run from the same directory where the database is located.

See *dbclean(1)* for more details.

The statistics produced by *dbcheck(8)*, mentioned above, can be used to determine whether you want to run *dbclean(1)* or not.

This decision is usually based on available time and disk space.

If you use *dbclean(1)*, **all** indexes become outdated, although an attempt is made to regenerate any defined autoindexes.

Database Recovery

dbrestor

FirstBase provides a mechanism to restore a database that gets destroyed during a hardware failure.

The recovery part of the mechanism takes a previously backed up database along with a current database log and creates a new database.

See *dbdump(8)* and *dbrestor(8)* for more details.

Environment Control Hints

All the *FirstBase* tools will follow some environment commands that can change the behavior of some or all of the *FirstBase* tools.

For example, there is an environment variable, `DIRNAME`, that if defined to be **ON** will place the current directory name on the footer line that is displayed on line 23 during most *FirstBase* tools.

setup

These environment variables are all set and controlled by the setup file. Alternately, they can be used on the command line of *FirstBase* commands.

There are many of these environment variables. For a complete list, and more details, see the manual page *setup(5)*.

Other Points Of Interest

There are other *FirstBase* tools that may be of some interest: *dbls(1)* provides an object oriented file listing; *dbpasswd(1)* provides rudimentary password locking of a database; and *dbvemit(1)* is a *dbvedit(1)* front end that only allows record creations and stores its results into a standard UNIX file.

There are many other fine points of interest concerning the *FirstBase* as a whole. The manual pages are very thorough in their discussion of these various nuances.

In particular, see the *coname(5)*, *generators(5)*, *input(5)*, and *screens(5)*. There are many others as well.

Summary

UNIX managers are usually expected to be fluent in manual ('man') pages. It is suggested that you eventually read all of the manual pages provided with *FirstBase*.

Section 5 and section 8 manual pages may be particularly interesting for *FirstBase* managers.

The reasonable man adapts himself to the world; the unreasonable one persists in trying to adapt the world to himself. Therefore, all progress depends on the unreasonable man.

Reason
GEORGE BERNARD SHAW

*pity this busy monster, manunkind, not.
Progress is a comfortable disease.*

E. E. CUMMINGS

*Not to go back, is somewhat to advance.
And men must walk at least before they dance.*

To Lord Bolingbroke, 1.53
ALEXANDER POPE

All that is human must retrograde if it does not advance.

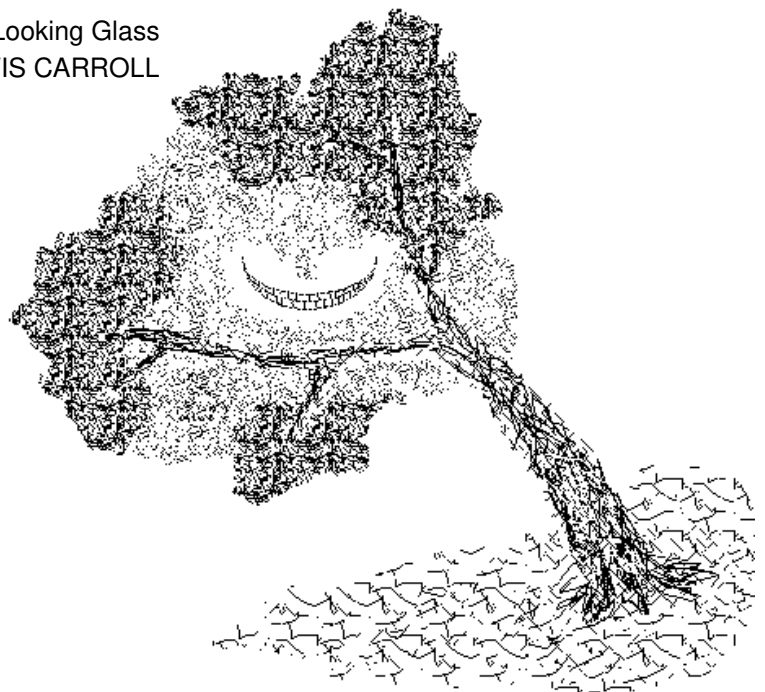
The Decline and Fall of the Roman Empire
EDWARD GIBBON

Truly great madness cannot be achieved without significant intelligence

Henrix Tikkanen

'The question is,' said Humpty Dumpty, 'which is to be master — that's all.'

Through the Looking Glass
LEWIS CARROLL



Advanced FirstBase Topics

This chapter covers advanced topics within FirstBase. These issues include more advanced uses of *dbshell*, *dbvedit*, *dbmacro*, and *dbawk*. Additionally, tips on application menus, intraline editing, view dictionaries, and trigger fields are also described.

EZ FirstBase

A set of menus and shell scripts that interactively prompt the user for object names while explaining the full options for each tool are located under the MAIN menu as selection **ezfb**.

These menus and scripts can be used from standard ASCII terminals since they are not window based.

Application Menus

A good method for creating robust menus of applications consist of defining a true UNIX environment variable for the home directory of the application and then using that variable in the *dbshell* menus.

If an application is called *master*, perhaps the environment variable MASTERHOME would be used.

In this case, the selections in the *dbshell* menu could then use the UNIX environment variable MASTERHOME within the arguments to *FirstBase* tools.

For example, the following shell script fragment would work and allow users to run the application from any directory at all:

```
dbvedit -d $MASTERHOME/master \  
        -i $MASTERHOME/coname \  
        -v $MASTERHOME/update
```

One reason for building applications in this manner is that each user can then be in their own directory, perhaps even using some of their own files to build parts of the record, or storing printouts without fear of conflict in the multi-user UNIX environment.

For more details on the building of menus, see Chapter 17, "Creating FirstBase Menus" and the manual page *dbshell*(1).

Intraline Editing

FirstBase has two types of data input, simple and editable. The simple type of input is the default method.

To enable the editable input, you must set the *FirstBase* environment variable EDITINPUT to ON as defined in *setup(5)*.

When editable input is enabled, all *FirstBase* uses of the input mechanism will allow keystrokes to correct existing data in place without erasing and retyping the entire input field.

Editable input is very handy for text fields within the *FirstBase* database editor — corrections far inside a field can be made with very few keystrokes.

The set of keystrokes used for editable input is extensible and can be redefined by each user. This keyboard mapping is done in via the *.firstbase-kbmap* file outlined in *keyboard(5)*. The input mechanism is described in *input(5)*.

The default key bindings released by FirstBase Software are an attempt to match a set of bindings prevalent in the editing world, including GNU Emacs and FrameMaker.

However, it is possible to redefine the keystrokes so as to make the *FirstBase* editors and tools respond to familiar keystrokes.

FirstBase languages

FirstBase supports a few languages or tools that can be used programmatically with *FirstBase* to extract and format data.

Standard *awk* Mechanism – *dbawk*

First, there is a standard *awk* mechanism called *dbawk(1)*. With it, you can write standard *awk* scripts that talk about database fields. Like *awk*, the looping over each referenced or indexed record is handled by the internal *dbawk* mechanism.

Macro Language – *dbmacro*

Another programmable language tool within *FirstBase* is *dbmacro(1)*. This tool interprets C like code fragments and comes complete with a large set of functions for string, numeric and date manipulations.

This tool can be used to generate printouts, do data analysis, or create files to be fed to other UNIX tools.

Additionally, the FirstBase macro language can be used to modify the behavior of the database editor, *dbvedit(1)*. See "View Dictionary Hints" on page 19-3.

Structured Query Language – *dbsql*

A structured query language tool is also provided. This tool, *dbsql*(1), supports the ANSI standard SQL constructs, with some extensions, to provide the common data manipulation language.

One powerful feature of *dbsql* is the *FirstBase* extension that allows the creation of standard *FirstBase* indexes from an SQL query. See the manual page for more details.

View Dictionary Hints

The *FirstBase* tool for visually editing databases is *dbvedit*(1). Full details on creating view dictionaries are located in *view*(5). A few useful hints for building robust view dictionaries are presented here. It is assumed you are already a bit familiar with view dictionaries.

Text Fields

When a large text field is defined within a *FirstBase* database, it is usually edited using a UNIX system editor such as *vi* or *emacs*.

Often, it is desirable to see a large section of the text on the screen within the database editor. The following view dictionary fragment shows how to do this task and still allow smooth interface into the UNIX editors:

```
"Text Field":10,1
textfield@11,1
textfield[1:79]@11,-1
textfield[2:79]@12,-1
textfield[3:79]@13,-1
```

A few points to notice: the editable point, line two, maps to row eleven, column one, exactly like line three, the display of the first line of text within the text field *textfield*.

Also, all displays of *textfield*, other than the editable point, are done using the display-only mark of a negative row/column component.

Macro Fields

FirstBase supports the extensible definition of the database editor via the *Firstbase* macro language. Fragments of macro code can be tied to individual fields to be executed at edit time, or even as the database record is being written.

This macro code can define complex defaults, ensure certain data conditions, or even ask for input confirmation.

Additionally, macro fields can be used to provide trigger fields. See the manual page for *macro*(5) for more details.

Trigger Fields

A trigger field is a field that conditionally controls the behavior of the database editor. These fields are macro fields that use programming constructs to control the trigger events.

When a trigger is needed, the best method is to use an additional field as the actual trigger point. The macro is tied to this field within the database dictionary.

The fields that the macro itself edits, i.e. the fields that are edited when the trigger is executed, should be defined as display-only in the database view dictionary.

Only the trigger point is editable to the view dictionary. Additionally, this field should be mapped to an unused location on the screen as it will cause a single blank to appear wherever it is drawn.

Here is a view dictionary fragment

```
FixSWTRIG:13,14
FixSW@14-2
FixSWVer@14,-30
```

Then, within the actual macro, the following can be used

```
editfield(FixSW)
editfield(FixSWVer)
```

This code will edit the two fields from within the macro even though the view dictionary has these fields as display only.

However, to get proper behavior from the database editor, many other signals (keystrokes) need to be covered. So, while the above code will work, better examples can be found in the \$FIRSTBASEHOME directory, in *applications/sample_macro*.

Applications

A wide array of applications can be found in \$FIRSTBASEHOME in the *applications* directory. These applications can be copied off and modified, used as is, or merely examined for ideas.

FirstBase Installation

This document describes how to install *FirstBase*, set up fixed and floating *FirstBase* licenses, and install new *FirstBase* users.

You will need about 12 - 15 MB of disk space depending on machine architecture and *FirstBase* options.

Installing *FirstBase*

All of the *FirstBase* tools and scripts are located in a single directory that can be situated anywhere on the UNIX file system.

The directory chosen will be referred to here as *install_dir*, though you should pick a name like *'/usr/local/firstbase'* or something.

Here are the steps for *FirstBase* installation:

mkdir

- (1) Make a directory for the *FirstBase* files:

```
mkdir install_dir
```

A couple of good places to install *FirstBase* might be something like */usr/local/firstbase* or */var/firstbase*. For these instructions, we will call this directory *install_dir*.

cd

- (2) Change to that directory.

```
cd install_dir
```

tar

- (3) Use the *tar(1)* command to extract the files from the tape.

```
tar fpx /dev/rst8
```

This command assumes the tape drive device is *rst8*. Your tape device could be *rmt0*, *st0*, or even *tape*. If you are unsure, see the system manager.

cp

- (4) Copy any previously installed *FirstBase* license information.

```
cp old_install_dir/*.lic* install_dir
```

- (5) Set up new *FirstBase* users.

To set up new *FirstBase* users, each user will need to either run the script *fbnewuser* from the *install_dir*, or modify their login environment.

fbnewuser

Once the environment is modified, remember to log out and then back in to make these changes take effect.

This setting should be done by the installer at this point, also.

NOTE

To enable *Global* license installation (in the next step), this step needs to be done by the installer *right now*.

The script *fbnewuser* will properly modify a *.cshrc* file for those using *csh(1)* or a *.profile* file for those using *sh(1)*. If you have both of these in your \$HOME area, you will have to choose which one you want to modify.

If you use *fbnewuser*, move on to the next step.

Here are the lines added by *fbnewuser* if you want to edit your *.cshrc* file by hand (for *csh(1)* users):

```
setenv FIRSTBASEHOME install_dir
set path=( $path \
$FIRSTBASEHOME/`arch`/bin \
$FIRSTBASEHOME/scripts \
$FIRSTBASEHOME/scripts_ez )
```

For those who are using *sh(1)* and want to edit the *.profile* file by hand, here are the needed lines:

```
FIRSTBASEHOME=install_dir
PATH=: $PATH:\
$FIRSTBASEHOME/`arch`/bin:\
$FIRSTBASEHOME/scripts:\
$FIRSTBASEHOME/scripts_ez:
export FIRSTBASEHOME PATH
FIRSTBASEHOME PATH
```

Note that these lines use the *arch(1)* command found on many UNIX operating systems. If your system does not have the *arch* command, then either create one that echoes the architecture name, or the directory will have to be hardwired, as in *hp720/bin*.

(6) Set up new *FirstBase* licenses.

There are two types of *FirstBase* licenses: fixed and floating.

Fixed licenses are tied to a single fixed CPU using the Host ID chip on the CPU or a small plug in Host ID device provided by FirstBase. The *FirstBase* software must then be used from this fixed node.

Floating licenses use a fixed CPU as a license server, but allow the *FirstBase* software to be executed from any CPU that has NFS access (using RPC - Remote Procedure Call) to the license server CPU.

fbfixed

OR

fbfloat

Before starting this procedure, contact FirstBase Software at license@firstbase.com or (520) 742-7897 to receive some or all of the following information: Support Date, License Type and License Password(s).

For fixed CPU licenses, be prepared to provide the Host ID of each CPU that will need to be licensed. This may entail plugging in a Host ID device provided by FirstBase Software. For Floating licenses, only the Host ID for the license server is required.

Once you have all of this information you are ready to begin.

Important: Your password is good only for the Host ID you provide to FirstBase Software customer service and is not transferable to another CPU.

To set up a new fixed CPU license for *FirstBase*, run the command *fbfixed* from *install_dir*.

For these fixed licenses, you will input the *hostid* for each CPU that will be running *FirstBase*, and the License Password received from FirstBase Software.

To set up a new floating CPU license for *FirstBase*, run the command *fbfloat* from *install_dir*.

For the floating licenses, you will input a sequence number, optional UNIX user and group ID numbers, and the floating license password received from FirstBase Software.

Setting up a *FirstBase* license is an easy task using a full screen, interactive license editor.

Merely follow the prompts and enter the requested information. More complete instructions on the use of the license editors are outlined in the following sections.

If you are using *FirstBase* floating licenses, be sure to modify the environment variable that needs changing. See "More On Floating FirstBase Licenses" on page A-5.

More On Fixed CPU *FirstBase* Licenses

FirstBase fixed licenses are used to register each *hostid* CPU that is running tools from the *FirstBase*. Each license adheres to a single CPU and is not limited by the number of *FirstBase* users.

This section describes the use of the *fbfixed* tool to add or modify *FirstBase* licenses.

To start the procedure, run the *fbfixed* command from the installation directory, *install_dir*. This will start up a license editor tool, a *FirstBase* tool that uses the normal *FirstBase* input mechanisms and keystrokes.

The following is a list of the items *fbfixed* will prompt for.

After each selection, remember to hit a <RETURN>.

Local or Global

This selection controls the location of the license password file that is being edited. Global licenses are stored in the *FirstBase* home directory, in this case *install_dir*.

Alternately, local licenses can be stored in the HOME directory of *FirstBase* users.

Probably **G** (or **g**) for global is the best choice, though if you have many *hostid*'s listed in the global file, storing local license passwords might be desirable to decrease the license lookup time on execution of each *FirstBase* tool.

During execution of any *FirstBase* tool, the local license file is searched. if it exists. If not, the global license file is used. Duplicating entries between global and local license password files is acceptable.

Support Date

This date is assigned to your installation site by FirstBase Software.

It can be entered as MM-DD-YY, MMDDYY, or MM/DD/YY.

For example, if the support date is July 15, 1990, the keystrokes

07/15/90

can be used to enter this date.

License Type

The license type controls the classification of license given to you by FirstBase Software. There are four types of licenses, each selected by the first letter of its name: Permanent, Secure and Temporary, and Limited.

Host ID

Once the above three items of information have been entered, *fbfixed* asks for one or more Host ID / License Password pairs of information.

As many pairs as needed can be entered during this phase — all of them will be merged with the license file upon completion.

The Host ID is the information returned from the command *hostid(1)*.

If you are just entering a single machine, the default Host ID can be used by entering a <RETURN>.

This default is the Host ID of the CPU being used.

Fixed License Password

The license password is a number assigned by FirstBase Software.

Once entered and merged with the license password file, *FirstBase* will be enabled.

Fixed License Completion

When done entering new passwords, respond with an **n** to the Add More Host ID's question, and the password merging will begin.

Once done, the screen pauses with a completion message. Hit one more <RETURN>, the screen will be cleared, and the licensing task is complete.

More On Floating *FirstBase* Licenses

FirstBase floating licenses are used to enable many CPUs to share a set of licenses. The licenses are provided via NFS using RPC (Remote Procedure Calls). All of the licenses for a particular network reside on a single CPU, in a floating license password file.

This section describes the use of the *fbfloat* tool to add or modify *FirstBase* floating licenses.

To start the procedure, run the *fbfloat* command from the installation directory, *install_dir*. This will start up a license editor tool, a *FirstBase* tool that uses the normal *FirstBase* input mechanisms and key-strokes.

The following is a list of the items *fbfloat* will prompt for. After each selection, remember to hit a <RETURN>.

Sequence Number

The Sequence Number is the positional number of floating license. The first floating license installed is sequence number one (1). The next is number two (2).

This selection will default to the correct number based on how many licenses are already in place.

If you are updating (or correcting) an existing license, you will have to input the proper sequence number. This can be accomplished by examining the *.firstbase-flicense* file to determine the sequence number of the license password that needs editing.

Exclusive User ID

The Exclusive User ID is a standard UNIX user ID number as defined in the */etc/passwd* file. If set, this floating license will be reserved for the UNIX user with this ID. Use a <RETURN> to allow all users.

Exclusive Group ID

The Exclusive Group ID is a standard UNIX group ID number as defined in the */etc/passwd* file. If set, this floating license will be reserved for the UNIX group with this ID. Use <RETURN> otherwise.

Floating License Password

The license password is a long sequence of numbers and letters given to you by FirstBase Software. Carefully, enter the license password, exactly as given to you.

Floating License Completion

When done entering new passwords, respond with an **y** to the Add This License question, and the password merging will begin.

Once done, the screen pauses with a completion message. A <RETURN> will continue to the next prompt.

If you have more floating licenses to install, hit another <RETURN> at the Add More Licenses prompt, and repeat the cycle for the next sequential floating license. Otherwise hit an **n**.

Once the editing is done, you must modify the *FirstBase* environment variable *LSERVER* in the *FirstBase* initialization file. This file is located in *install_dir* and is named *.firstbase-init*.

Change this variable to be the host name of the license server, and remove the comment marker (#) from the front of the line.

Additionally, the *FirstBase* license server, *fbserver*, should be running. See the manual pages *setup(5)* and *fbserver(8)* for more details.

Testing The Installation

Once *FirstBase* is installed, the commands *firstbase* and *dbshell* should start the *FirstBase* menu shell using the MAIN menu. Additionally, the *fb* command will do the same thing.

fbdemotool

The top and next to last lines should be reverse video, with a nice, ordered display of selections in the middle of the screen.

OR

If OpenLook or SunView is being used, *dbshelltool* will provide a button overlay of the MAIN menu.

fbdemo

Other commands to try are *fbdemo* and *fbdemotool*, menu front ends to the pre-existing applications.

Still yet another command is *firstbasetool*, a *FirstBase* environment for use with Open Look.

For floating license users, each of the above commands will automatically request a license from the server and return it when done using the commands *fbgetlicense* and *fbputlicense*.

Internal Notes

- The file *install_dir/SEQF* must be writable by ALL who will create *FirstBase* databases. Although written to the tape correctly, this is a point that could cause problems if the permissions are not correctly set.
- When using a machine with NFS (like Sun Microsystems), you must be running the lock daemon, */etc/rpc.lockd* (or using the *FirstBase* lock daemon, *fblockd(8)*).

Use **ps -ax** to see if your machine is running this background daemon. See *lockd(8)* if you need more details.

If *lockd(8)* is not running, *FirstBase* will create a hung process which can only be killed by starting */etc/rpc.lockd*, issuing a **kill -9** signal, and waiting 60 seconds.

NOTE

If you are running NFS across machines using combinations of SunOS 4.0.3c and SunOS 4.1, there has been a ninth patch released by Sun (100075-09) covering the *lockd* discrepancies.

Contact Sun or FirstBase if you have any *lockd* problems.

- See the file **README** in *install_dir* for late-breaking news and topics concerning new features in the *FirstBase* system.

If you need any assistance, please feel free to call us and ask questions.

If you are on the Internet, you can even send electronic mail.

Enjoy!

FirstBase Software
(520) 742-7897
firstbase@firstbase.com

Steps for Defining a FirstBase Database

This is a summary of the steps involved in creating a *FirstBase* database dictionary. These steps are explained in much greater detail in Chapter 4, "Defining a FirstBase Database".

The following commands are given to the define database tool, formally called *dbdbas(1)*.

- (1) Use @<RET> to enter auto-add mode.
- (2) *Field*: The maximum field name length is 10 characters. These must be letters or numbers, no punctuation marks are allowed.
- (3) *Type*: default is "a" (alphanumeric): See *dbdbdas(1)* for a complete list of field types.
- (4) *Size*: default is 10 characters: There is no upper limit to field size. However, some form of text editor must be used when the field size is greater than 300 characters.
- (5) *Default*: default is "forced entry": The text entered here will be defaulted to during *dbedit(1)* if no other data is chosen. Entering a blank here means that there is no default and no forced entry.
- (6) *Comment* (optional): A user-defined comment which will accompany the field may be entered here. Max. comment size is 50 characters.
- (7) *C.Loc.*: (Comment Location) is used only when Comment is defined. a = after the field, b = before the field
- (8) *Lock?*: default is no: Entering y(es) here prevents the field from being edited after initial data entry.
- (9) *Change Help* (optional): Entering y here allows the creation of a user-named help file that may be called from this field.
- (10) *Change Choice* (or *Extended Choice*): Enter y and the name of the user-created choice file that will be referenced from this field. The choice file should exist and be named here before using *dbedit(1)*.
- (11) *Change Auto Index Info?*: Entering y here allows the creation of a user-named autoindex, which is indexed and sorted by this field. (All database records can be quickly located with an autoindex.)

- (12) *Change Valid Data Range*: Entering **y** here allows you to specify the range of data values that will be accepted during data entry. The data range can be specified in the form xxx-yyy or a,b,c. A Date field range is specified as YYMMDD.
- (13) *Change Macro File Name*: Entering a **y** here allows you to specify a macro file that will be executed during the use of the database editor *dbvedit*. The macro file can assign a default value or even provide the entrance to conditional editing of other fields. See *macro(5)* for more.
- (14) Repeat steps 2-13 for each field in your database dictionary.
- (15) When you are done adding fields to your database, enter an END keystroke, “-”, in the Field location. Field changes can be made at this time if needed. Another end keystroke will generate the database dictionary and return you to the *FirstBase* Main Menu.
- (16) If you decide not to keep the dictionary, use the abort keystroke, “CTL-X”.

Conversion to *FirstBase*

FirstBase was originally released as the *Cdb Toolkit*. This document describes how to convert a *Cdb Toolkit* installation to *FirstBase*.

The procedures here are fairly easy, and mostly involve the renaming of *FirstBase* system files, and the modification of a UNIX environment variable.

Cdb Toolkit to FirstBase

The following list assumes a working knowledge of UNIX as the actual keystrokes are not given, merely a list changes to be made when converting *Cdb Toolkit* sites over to *FirstBase*.

- move any *.cdbrc* files to *.firstbase-init* files
- move *.cdblicense* files to *.firstbase-license* files
- move *.cdbflicense* files to *.firstbase-flicense* files
- move *.cdb_kbmap* files to *.firstbase-kbmap* files
- change environment variable CDBHOME to FIRSTBASEHOME.
This is the true UNIX environment variable from the *.cshrc* or *.profile* file.

Other *FirstBase* Modifications

Other changes mostly affect the names of tools. Here is a summary:

- *cdbtool* has been renamed *dbshelltool*
- All other *cdb****** tools are now named *fb******
- *rcdb* has been renamed *rfb*
- the *libcdb.a* file (for library users) is now *libfb.a* so that instead of using the **-lcdb** *ld(1)* options, the **-lib** library should be used.

Index

A

- Add Mode 5-6
- adding records 5-6
- alphanumeric - a 5-15
- Any Change screen 2-10
- autoindex 4-8, 8-11

B

- backup 18-3
- basic input mode 2-5
- binary - b 5-15
- Btree index 8-11, 8-12

C

- case insensitive searching 5-10
- changing field size 12-2
- changing field type 12-2
- choice - c 5-15
- Choose Field screen 2-12
- clean database 18-2
- command line 2-8
- common screen layout 2-7
- common screens 2-10
- communicating with FirstBase 2-1
- complex autoindex 8-11
- conversion methods 12-8
- convert database 12-1
- correcting data 2-5

D

- data dictionary 4-3
- data entry 5-1
- database
 - concatenation 13-1
 - conversion 12-6
 - creation 4-9
 - generation 12-6
 - joining 14-1

- map 18-2
- move - mvdb(8) 12-8
- recovery 18-3
- remove - rmdb(8) 12-8
- update 15-1

- date - d 5-16

- dbawk(1) 7-8, 19-2

- dbcat(1) 13-1

- dbcgen(1) 12-1

- dbcheck(8) 18-2

- dbclean(8) 18-2

- dbdbas(1) 4-1

- dbdcnv(1) 12-1

- dbdind(1) 8-1

- dbdlbl(1) 9-1

- dbdmrg(1) 10-1

- dbdprt(1) 7-1

- dbdscr(1) 6-1

- dbdump(8) 18-3

- dbdupd(1) 15-1

- dbedit

- adding new records 5-6

- basic concepts 5-2

- command level mode 5-5

- command screen 5-3

- field level 5-13

- record level 5-11

- dbemit(1) 7-8, 16-1

- dbfilter(1) 8-13

- dbigen(1) 8-1

- dbjoin(1) 14-1

- dblgen(1) 9-1

- dbload(1) 16-1

- dbls(1) 18-3

- dbmacro(1) 7-8, 15-1, 19-2

- dbmerge(1) 10-1

- dbpack(1) 11-2

- dbpasswd(1) 18-3

- dbpgen(1) 7-1

- dbregen(1) 8-12

- dbrestor(8) 18-3

dbrload(1) 15-1
dbscan(1) 18-1
dbshell(1) 17-1
dbsql(1) 7-8, 19-2
dbugen(1) 15-1
dbundo(1) 11-2
dbvedit(1) 5-2, 6-4
dbvemit(1) 6-4, 18-3
dbvform(1) 6-4, 7-8
dbvi(1) 6-4
dbvscan(1) 18-1
default 4-12
define database 4-1, 4-3
define index 8-1
deleting a record 5-12
deleting records 15-4
document merging 10-1
dollar - \$ 5-16
dots 2-4
download database 16-1
dynamic join 14-1

E

easy-to-use firstbase 19-1
editable input mechanism 19-2
editable input mode 2-5
editor 5-1
emit values 16-1
END key 2-6
environment commands 3-3
environment control hints 18-3
extended choice 5-17
EZ FirstBase 19-1

F

field
 default 4-12
 naming 4-4
 type hints 4-11
 types 5-15

field default 4-12
field programming 5-13
file names 3-5
fixed license A-1
float - f 5-16
floating license A-1
formula - F 5-16
formula fields 5-13

G

global database update 15-1

H

help key 2-7

I

index
 And/Or column 8-10
 autoindex 4-8, 8-11
 Btree 8-11, 8-12
 create 8-2
 define 8-2
 field column 8-3
 generate 8-7
 move - mvidx(8) 8-11
 organization count 8-12
 record count 8-12
 remove - rmidx(8) 8-11
 sort by 8-5
 Val1 Column 8-3
 Val2 column 8-9
index definition 8-1
index generation 8-7
input dots 2-4
input modes 2-5
input(5) 19-2
installing FirstBase A-1
integrity 18-2
intraline editing 2-5, 19-2

J

joining databases 14-1

K

keyboard(5) 19-2

L

label generation 9-1

languages 19-2

license installation A-1

Link - L 5-17

link fields 14-1

load database 16-1

M

macro fields 5-13, 19-3

macro(5) 6-4

mail merge 10-1

mailing labels 9-1

MAIN menu 3-2

makeview(8) 6-5

manager tasks 18-1

menu creation 17-1

menu shell 3-1

multi user 18-1

N

non-indexed searching 5-11

nroff(1) 7-8

numeric - n 5-17

O

operating FirstBase 3-1

P

pattern matching 5-10

perl(1) 7-8

positive numeric - N 5-17

printing a report 7-7

printout 7-1

programmable fields 5-13

R

read-only editor 18-1

record locking 18-1

recoverable errors 2-9

recovery 18-3

REGEXP 5-10

regular expression 5-10

removing deleted records 11-1

report 7-1

report generation 7-5

reshape database 12-1

restoring deleted records 11-1

restructure database 12-1

S

scan 18-1

screen dictionary 6-2

screen footer 2-8

screen header 2-7

screenprint 7-7

screens 6-1

security 18-1

setup(5) 18-3

shell 17-1

silent choice - C 5-16

sort by fields 8-5

sort(1) 8-11

T

tbl(1) 7-8

terminal 2-1

text fields 19-3

trigger fields 19-4

types of fields 5-15

U

update 15-1
upload database 16-1
uppercase - U 5-17

V

view dictionary 6-4
view(5) 5-2, 6-4
viewing a report 7-7
views 6-1

Y

Yes/No Question screen 2-11