SAS Publishing

Getting Started with SYSTEM 2000® DBMS

An Introductory User's Guide

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 1987. *Getting Started with SYSTEM 2000® DBMS, An Introductory User's Guide*. Cary, NC: SAS Institute Inc.

Getting Started with SYSTEM 2000® DBMS, An Introductory User's Guide

ISBN 1-55544-100-9

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, 1987

SAS Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at support.sas.com/pubs or call 1-800-727-3228.

PREFACE

SYSTEM 2000 DBMS (Data Base Management System) is a product of SAS Institute Inc. This system operates on IBM, CDC, and Sperry mainframe computers.

This manual is for anyone who wants an introduction to SYSTEM 2000 DBMS. The explanations and examples are written in a way that lets you begin using SYSTEM 2000 DBMS immediately. You don't need any previous knowledge of the system, and you don't need to be a computer expert to try the examples.

The purpose of this manual is to acquaint you with the system and to demonstrate that you can easily learn to be productive using the system. It also gives you a taste of the richness of SYSTEM 2000 DBMS, illustrating some of its more powerful options. If you would like to understand more about using these and other options, you can read details in other SYSTEM 2000 manuals listed in Part 9.

Part 1 gives an overview of SYSTEM 2000 DBMS and sample commands with their results. Part 2 shows you how to get started using the system. Because the procedures for calling SYSTEM 2000 DBMS vary at each installation, you will need to ask your data base administrator for your unique procedure. Part 3 talks about retrieval commands, and Part 4 describes update commands that are available for adding and modifying data.

Part 5 introduces you to the concepts of data base structure, shows you how to define a data base, and gives you some simple commands for creating your own new data base. Part 6 describes how to store commands and functions that you use frequently.

Part 7 gives a brief description of additional facilities that are available with the system. Part 8 contains hints, time-savers, and guidelines for avoiding common errors. The topical index at the end of the manual will help you locate concepts, facilities, and commands described in this manual.

The EMPLOYEE data base is used for most of the examples. The data base definition and four sample logical entries are given in the fold-out at the end of the manual. This sample data base is included on your SYSTEM 2000 installation tape.

If you would like more literature about SYSTEM 2000 facilities, see the list of other manuals in Part 9. To obtain copies of these manuals, see your data base administrator or contact SAS Institute Inc. Also, use the SYSTEM 2000 Subject Index to locate discussions about specific topics.

TABLE OF CONTENTS

		page
PART 1:	INTRODUCTION	1-1
	1.1 Overview	1-1
	1.2 Excerpts from Sample Sessions	1-3
PART 2:	GETTING STARTED	2-1
	2.1 Format of Commands	2-1
	2.2 Beginning and Ending a Session	2-2
	2.3 Creating a New Data Base	2-4
PART 3:	RETRIEVING INFORMATION	3-1
	3.1 Displaying the Data Base Definition	3-2
	3.2 Tallying Key Values	3-5
	3.3 Printing Data Values	3-7
	3.4 Listing Values in Columns	3-10
	3.5 Selecting the Data You Want	3-14
	3.6 Sorting Your Output	3-17
	3.7 Repeating a Command	3-18
	3.8 Limiting Your Output	3-18
PART 4:	UPDATING DATA	4-1
	4.1 Adding Values to Existing Records	4-2
	4.2 Changing Values	4-3
	4.3 Removing Values	4-4
	4.4 Inserting New Records	4-5
	4.5 Removing Records	4-6
PART 5:	DEFINING AND LOADING A DATA BASE	5-1
	5.1 Terminology	5-1
	5.2 Defining a Data Base	5-4
	5.2.1 DEFINE Sessions	5-4
	5.2.2 Defining Items and Records	5-5
	item definitions	5-6
	record definitions	5-8
	5.2.3 Order of Definition	5-9
	5.3 Loading the Data	5-10
PART 6:	STORING FREQUENTLY USED COMMANDS	6-1
	6.1 String and Function Definitions	6-1
	6.2 Functions	6-2
	6.3 Parametric Strings and Functions	6-4

TABLE OF CONTENTS (continued)

page

PART 7: OTHER SYSTEM 2000 FEATURES 7-1
7.1 Self-Contained Facility Options 7-1
7.1.1 Calling the Various Languages 7-1
7.1.2 The CONTROL Language 7-2
7.1.3 The QUEUE Language 7-2
7.1.4 The CREATE Facility 7-2
7.1.5 User Files 7-3
7.2 The Genius Facility and the REPORT Language . . . 7-4
7.3 The QueX Facility 7-8
7.4 The Programming Language Extension (PLEX) Facility 7-16

PART 8: TIME-SAVERS AND GUIDELINES FOR AVOIDING ERRORS 8-1
8.1 Timesaving Techniques 8-1
8.2 Avoiding Errors 8-2
8.2.1 Syntax Problems 8-2
8.2.2 Unexpected Results 8-3
too much information 8-3
not enough information 8-4
ignored commands 8-4
incorrect deletions 8-4

PART 9: SYSTEM 2000 REFERENCE LITERATURE 9-1

APPENDIX A: SAMPLE FORTRAN PLEX PROGRAM A-1
A.1 General Structure of a PLEX Program A-1
A.2 Sample PLEX Program (DIRINT) A-3

INDEX X-1

PART 1: INTRODUCTION

Chapter 1.1: Overview

SYSTEM 2000 DBMS (Data Base Management System) is a product of SAS Institute Inc. This user's guide is for anyone who wants to try SYSTEM 2000 DBMS in a relaxed atmosphere, using simple SYSTEM 2000 commands to display and modify information stored in a data base.

SYSTEM 2000 DBMS allows you to define your data base, store the data, and display information in a variety of ways. The format of SYSTEM 2000 commands is similar to an English sentence. With a few simple rules, you can quickly learn to access the information you need at your desk terminal.

Each data base contains labeled items and records. **Items** contain your actual data values. **Records** associate related items. Records are also related to each other, some being subordinate to others. Item and record labels as well as the actual data values can be in any language, for example, French, English, or German.

This manual gives you enough information to use SYSTEM 2000 DBMS quickly. You can try the various retrieval and update commands shown in the examples. You can also create and try your own examples, following the patterns shown. However, be aware that some of your attempts might be unsuccessful, simply because your knowledge of the system is still limited.

Before we begin, it might help to give you an overview of the various SYSTEM 2000 languages. The **DEFINE**, **CREATE**, **QUEST**, **QUEUE**, **REPORT**, and **CONTROL** languages form the **Self-Contained Facility (SCF)**, so named because these languages can be used as needed without leaving the session.

To create or modify a SYSTEM 2000 data base definition, use the DEFINE language. Also, the CREATE feature is a question/answer type of language for defining a data base; it is available only for some versions of the system. To retrieve or update data, use the QUEST language. The REPORT language lets you produce specially formatted reports. With the CONTROL language you can make copies of your data base at any time and assign password security to the items and records in your data base. The QUEUE language queues many retrieval and update commands together in a batch for fast processing of volume applications.

This manual also introduces other SYSTEM 2000 facilities, such as Genius, QueX, and PLEX. You might want to learn more about these and other available features after you are comfortable with the Self-Contained Facility and SYSTEM 2000 data bases.

Genius produces reports and listings through a series of prompts and user responses. For example, Genius prompts you for titles, data to be displayed, and calculations to be performed. Then it uses your responses to produce the report listing. Genius automatically centers information on the pages and edits numeric data with commas and dollar signs as needed.

QueX is a menu-driven system. You retrieve and update your data by selecting an action from a menu displayed on your screen.

The Programming Language Extension (PLEX) feature is an alternative for users who want to access a SYSTEM 2000 data base through a programming language, such as COBOL, FORTRAN, PL/I, or Assembler. PLEX is used mainly by the programmer who needs to load or update large volumes of data.

The Genius, QueX, and PLEX facilities are described briefly in Part 7. For more details about any of these, ask your data base administrator for a copy of the specific SYSTEM 2000 manual describing the feature of interest to you.

Chapter 1.2: Excerpts from Sample Sessions

Before learning how to use SYSTEM 2000 DBMS, it might be helpful to look at some examples of SYSTEM 2000 commands and results. Don't be concerned with the details of these examples; they are explained in later chapters.

The following sample output is for a TALLY command that shows the department names and the number of times each name occurs in the EMPLOYEE data base.

```
TALLY DEPARTMENT:
**************************************************
ITEM-        DEPARTMENT
**************************************************
OCCURRENCES   VALUE
------------------------------------------
         8    ADMINISTRATION & FINANCE
        11    CORPORATION
        18    INFORMATION SYSTEMS
        21    MARKETING
------------------------------------------
         4 DISTINCT VALUES
------------------------------------------
        58 TOTAL OCCURRENCES
------------------------------------------
```

This LIST command displays the first names and last names of employees in the Marketing department.

```
LIST /TITLE D(7)MARKETING EMPLOYEES/ FORENAME, LAST NAME
WHERE DEPARTMENT EQ MARKETING:

  MARKETING EMPLOYEES
      12/15/1987

 * FORENAME              LAST NAME
 ***
 * OLAN M.               GARRETT
 * VIRGINA P.            BROWN
 * FANNIE                LITTLEJOHN
 * MADISON A.            SCHOLL
 * RUBEN R.              BROOKS
 * ARMANDO               JUAREZ
 * DAVID                 AMEER
 * TRAVIS Z.             RICHARDSON
 * MERRILEE D.           WAGGONNER
     .                     .
     .                     .
     .                     .
```

This PRINT command displays employee number, name, department, job, and salary data for employee number 1265.

```
PRINT EMPLOYEE NUMBER, LAST NAME, FORENAME, POSITION WITHIN COMPANY
    WHERE EMPLOYEE NUMBER EQ 1265:
 1*  1265
 2* SLYE
 3* LEONARD R.
   101* GENERAL MAINTENANCE
   102* ADMINISTRATION & FINANCE
   103* SQT
   104* SUPPORT
   105* 04/02/1979

     111*       $3.50
     112* HOURLY
     113* 04/02/1979
     114*      $55.73

       121* 05/31/1979
       122*   80.00
       123*   12.00
       124*     $343.00
       125*      $37.20
       126*     $288.27

       121* 04/30/1979
       122*   80.00
       123*     .00
       124*     $280.00
       125*        $.00
       126*     $263.20
```

This one lists the accumulated vacation and sick leave hours for the corporate employees.

```
LIST LAST NAME,(ACCRUED VACATION + ACCRUED SICK LEAVE)
     WHERE DEPARTMENT EQ CORPORATION:
 * LAST NAME
 ***
 * WATERHOUSE          16.000
 * SALAZAR            136.000
 * BOWMAN             120.000
 * NATHANIEL           48.000
 * GARRETT            140.000
 * KNIGHT              16.000
 * MILLSAP             72.000
 * FAULKNER            64.000
 * KNAPP               52.000
 * MUELLER             80.000
```

PART 2: GETTING STARTED

Chapter 2.1: Format of Commands

The Self-Contained Facility offers easy-to-use commands composed of keywords (recognized by the system) and words supplied by the user. Each command begins with a keyword that designates the type of operation the user is requesting. Keywords are uppercase and user-supplied words are lowercase in the command formats.

Each command ends with a special character, usually the colon or semicolon. For some versions, this command terminator is set at the customer site, and it can be, for example, a percent sign. Check with your data base administrator. Examples in this manual use the colon as the command terminator.

SYSTEM 2000 commands are free format; that is, leading, trailing, and extraneous embedded blanks are ignored. However, do not enter blanks in the middle of a word or the system will not recognize the word. For example, the keyword DESCRIBE would not be recognized if you enter DES CRIBE. (Of course, all blanks are retained for you in text titles and column headings.)

A SYSTEM 2000 data base has items and records. You define a number and a name for each of them. Then you can refer to them by either the name or the number in any command. Generally, the number is preceded by the letter C. This notation is known as the **C-number**, for example, C1, C21, C400.

SYSTEM 2000 commands have many options. Defaults are automatically set at the beginning of each Self-Contained Facility session for most of the options. For example, output from retrieval commands is single-spaced unless you specify otherwise.

SYSTEM 2000 DBMS checks each command and issues an error message if any part of the command is invalid. You can then reenter the command. If the error message is not complete in its description of the problem, you can refer to the SYSTEM 2000 Messages and Codes (MAC) manual for more help.

The SYSTEM 2000 Syntax Guide contains all Self-Contained Facility commands and numerous options not discussed here. It is a handy reference for learning more details about the Self-Contained Facility and its capabilities. Each command is illustrated with several examples.

Chapter 2.2: Beginning and Ending a Session

Getting started with SYSTEM 2000 DBMS is easy. You will need the following information to log on the computer, to call SYSTEM 2000 DBMS, and to log off the computer. Ask your data base administrator for this site-dependent information, particularly whether the command terminator is the colon, semicolon, or percent sign. Remember this manual uses the colon. Jot these down in the spaces below.

_____ your logon identifier

_____ User/Account information

_____ key or procedure for calling SYSTEM 2000 DBMS

_____ logoff key or keyword

_____ SYSTEM 2000 command terminator

After you have called SYSTEM 2000 DBMS, enter your password for the data base you wish to access by giving the following command.

USER, password:

Then, enter the **DATA BASE NAME IS command**, giving the name of the data base you wish to access. DBN IS can be used as an abbreviation for the DATA BASE NAME IS command.

DBN IS data base name:

Next enter the **ECHO ON command**. ECHO ON asks SYSTEM 2000 DBMS to display every line of input that you enter.

ECHO ON:

The echoes are very helpful if you have entered any word incorrectly or if transmission problems occur. The system issues an error message for words it can't recognize and points to the problem area. See Part 8 for hints about avoiding errors.

Now that you have opened the data base, and turned on the echoes, you can enter retrieval or update commands.

When you are ready to exit from SYSTEM 2000 DBMS, enter

EXIT:

If you want to log off the computer after finishing your Self-Contained Facility session, enter your logoff key. Otherwise, you can call SYSTEM 2000 DBMS again or use other utilities available at your site.

Here is a sample Self-Contained Facility session. These commands log on the computer, access an existing data base, and log off the computer.

IDME	site-dependent logon key
DEPT45	site-dependent User/Account
S2K.	site-dependent call to SYSTEM 2000 DBMS
USER, DEMO:	data base password
DBN IS EMPLOYEE:	name of data base to be accessed
ECHO ON:	command to turn on echoes
DESCRIBE:	sample retrieval command
.	
.	SYSTEM 2000 retrievals/updates
.	
EXIT:	exit from SYSTEM 2000 DBMS
OFF	site-dependent logoff of computer

During a Self-Contained Facility session, you can try any of the commands discussed in this manual. You can also read other manuals for more detailed explanations of each language. Part 9 lists the SYSTEM 2000 manuals.

Chapter 2.3: Creating a New Data Base

The procedure for creating a new data base is the same as accessing an existing data base with the following differences.

1. Choose a name for your new data base. Each data base name must be unique, so check with your data base administrator to make sure that the name does not duplicate any existing data base. Also, tell your data base administrator the name so he can set up the new data base files on the system.

2. Enter the **USER command**, which begins every session. When you are creating a new data base, the password you give becomes the **master password** for the new data base. Choose any word of 1 to 4 characters. The master password holder has access to all commands and all data in the data base. (With the CONTROL language, you can assign secondary passwords, which have limited access to the data base.)

3. Enter the **NEW DATA BASE IS command** to specify your new data base name (rather than the DBN IS command). You can abbreviate this command to **NDB IS** .

   After you name your new data base, SYSTEM 2000 DBMS gives you the DEFINE language to enter your new item and record definitions.

4. Enter your new item and record definitions. Chapter 5.2 tells you how to define items and records.

5. Enter the **MAP command** to tell SYSTEM 2000 DBMS to process your definitions and store them in your new data base.

   The MAP command stores your new definition (or modifies an existing definition). SYSTEM 2000 DBMS then gives you the QUEST language.

6. Enter the **DESCRIBE command** to display your data base definition. If you need to make corrections, call the DEFINE language again with the DEFINE command. You can change a data base definition at any time; however, it is easier to modify it before you load any data.

Here is a sample session to create a new data base. Make sure you check the data base name (and files) with your data base administrator before beginning a session.

```
IDME                          site-dependent logon key
DEPT45                        site-dependent User/Account
S2K.                          calls SYSTEM 2000 DBMS
USER, MINE:                   specifies master password
NDB IS MAILIST:               names new data base
ECHO ON:                      turns on echoes of commands
1* NAME (CHAR X(40)):         defines an item
2* ADDRESS (CHAR X(50)):      defines an item
   .
   .                          other item and record definitions
   .
MAP:                          stores the definition
DESCRIBE:                     displays the definition
   .
   .                          other SYSTEM 2000 commands, if any
   .
EXIT:                         exits from SYSTEM 2000 DBMS
OFF                           site-dependent logoff of computer
```

After you have created your new data base definition, you can load some sample data and then try a few retrieval and update commands, which are described in this manual.

Note: the CREATE facility might also be available at your site. CREATE allows you to define and load a data base conversationally through a series of prompts and user responses.

PART 3: RETRIEVING INFORMATION

The **QUEST language** is easy and effective. With the commands illustrated in this part, you can quickly learn to access information and browse through a data base. Here are some of the most frequently used QUEST retrieval commands.

DESCRIBE displays the data base definition.

TALLY displays and counts values for key items.

PRINT displays values vertically, one item per line.

LIST displays values in columns across the page, one item per column.

You have many options with each of these four basic commands. You can select a subset of data records to be displayed in the PRINT or LIST commands. Also, you can sort your output.

Other SYSTEM 2000 facilities, such as QueX, Genius, and PLEX, can also be used for retrieving data. Part 7 gives an overview of these features. Refer to other SYSTEM 2000 manuals for more details.

The first four chapters of this part show you how to use the DESCRIBE, TALLY, PRINT, and LIST commands. The last four chapters talk about selecting your data, sorting data, using the DITTO command, and controlling the amount of output with the LIMIT command.

Chapter 3.1: Displaying the Data Base Definition

A data base definition is created by the user and stored in the data base with DEFINE language commands. That definition is the key to all communication between you and SYSTEM 2000 DBMS. It contains the labels and attributes for the items, along with record definitions and relationships.

The **DESCRIBE command** shows you the data base definition; it does not display data. Use this command to find the names or numbers of items or records that interest you. The DESCRIBE command is also useful for looking at the types of data stored in the data base. SYSTEM 2000 item types include

CHAR	for alphanumeric data
INTEGER	for whole numbers
DECIMAL	for numbers containing a decimal point
MONEY	for decimal numbers having a dollar sign
DATE	for dates.

Note: some versions of SYSTEM 2000 DBMS allow other special item types, such as TEXT, OCTAL, BINARY, REAL, and ARRAY.

The format of a DESCRIBE command can be any of the following. Component labels in the command are C-numbers or names of items or records.

DESCRIBE:	to see entire definition
DESCRIBE component:	to see one item or record
DESCRIBE component THRU component:	to see a series of items

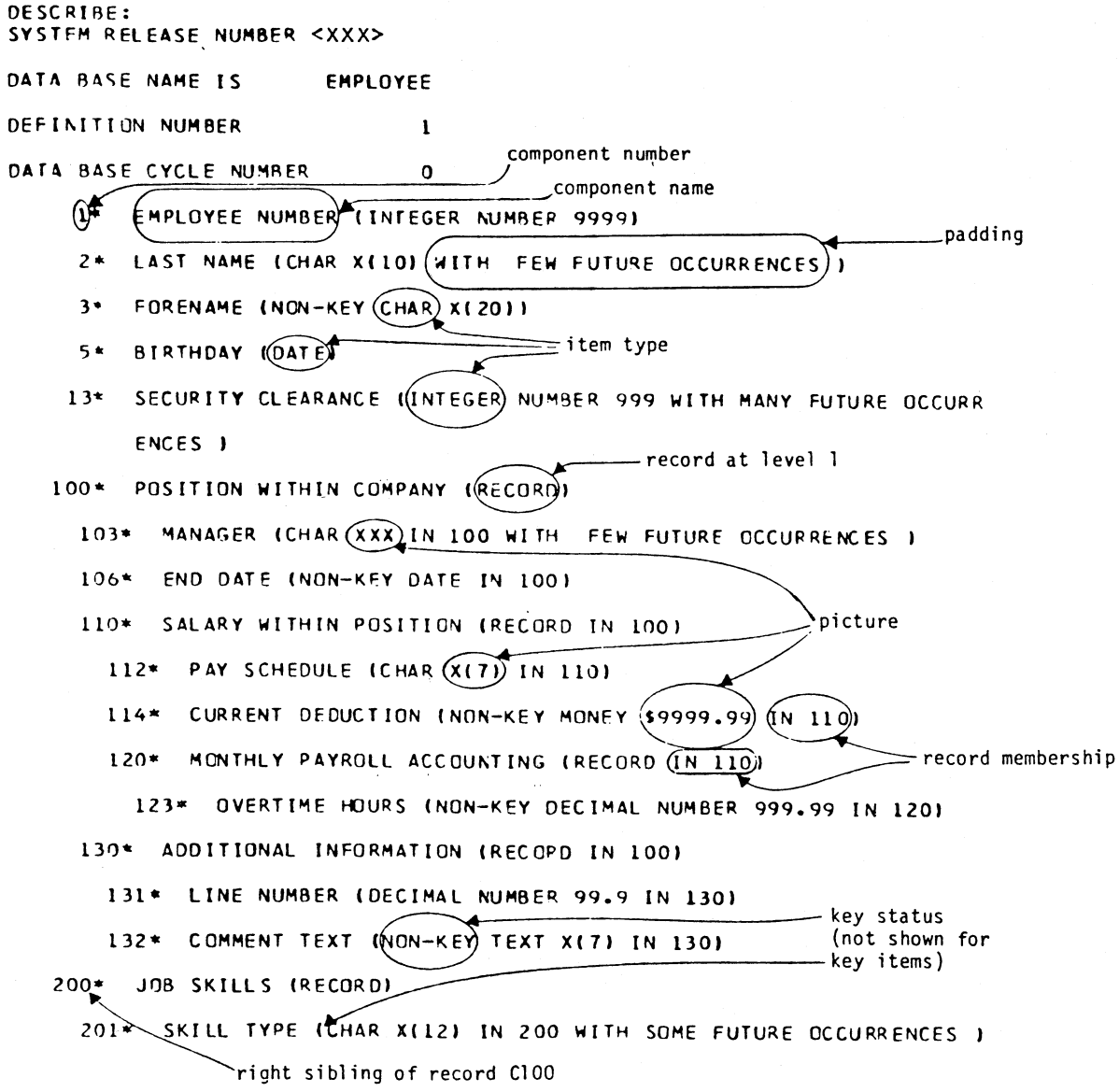
You can also store strings and functions in the data base definition. Strings contain commands or parts of commands that can be called without reentering them at your keyboard. Stored functions contain frequently used arithmetic calculations. You can define, delete, or modify stored strings and functions any time with the DEFINE language. To look at the strings or functions in a data base, enter one of these commands:

DESCRIBE STRINGS:
DESCRIBE FUNCTIONS:

The following examples show some sample output from the DESCRIBE command. Notice that for a full description the heading contains the definition version number and the data cycle number. These numbers show you how many times the definition has been modified and how many updates have been performed on the data.

This is part of the DESCRIBE output for the EMPLOYEE data base; the complete definition is in the fold-out at the end of this book. Annotated terms are explained in detail as we go along in the manual.

DESCRIBE Output



These examples illustrate the DESCRIBE command for a single item. You can define any item as key or non-key, that is, indexed (for fast access) or not indexed.

The item described in the first example stores integers having a maximum of four digits. By default, the item is indexed (key) for fast retrieval. The second item (C6) stores nonnumeric values that are normally no longer than eleven characters. This item is non-key; that is, it is not indexed.

```
DESCRIBE EMPLOYEE NUMBER:
    1* EMPLOYEE NUMBER (INTEGER NUMBER 9999)

DESCRIBE C6:
    6* SOCIAL SECURITY NUMBER (NON-KEY CHAR X(11))
```

The following example illustrates the DESCRIBE command for a record. Notice that only the record definition is displayed, not the items in the record.

```
DESCRIBE C110:
   110*  SALARY WITHIN POSITION (RECORD IN 100)
```

The last example illustrates displaying a series of components using THRU. The first component specified in the command must appear in the data base definition before the second component specified.

```
DESCRIBE C300 THRU C420:

300*  PERSONAL INTERESTS (RECORD)
  301*  INTEREST (CHAR X(12) IN 300 WITH FEW FUTURE OCCURRENCES)
  302*  AFFILIATION (NON-KEY CHAR X(5) IN 300)
  303*  COMMENT (NON-KEY TEXT X(5) IN 300)
400*  EDUCATIONAL BACKGROUND (RECORD)
  410*  EDUCATION (RECORD IN 400)
    411*  SCHOOL (CHAR X(15) IN 410)
    412*  DEGREE/CERTIFICATE (CHAR X(7) IN 410 WITH FEW
          FUTURE OCCURRENCES)
    413*  DATE COMPLETED (DATE IN 410)
    414*  MAJOR FIELD (NON-KEY CHAR X(16) IN 410)
    415*  MINOR FIELD (NON-KEY CHAR X(12) IN 410)
  420*  TRAINING (RECORD IN 400)
```

Chapter 3.2: Tallying Key Values

The **TALLY** (or **TA**) command gives a count of the distinct values for a key item, the number of times each value occurs, and the minimum and maximum values. It also displays the values in sorted order (ascending).

These are the formats for the TALLY command.

```
TALLY key item:
TALLY key item, key item ... :
TALLY/EACH/ key item ... :
TALLY/ALL/ key item ... :
```

Note: some versions of SYSTEM 2000 DBMS allow you to perform tallies on a subset of the data base (TALLY with a where-clause). See your data base administrator to find out if this option is available at your site.

The following examples show different forms of the TALLY command, using the EMPLOYEE data base. When experimenting with the TALLY command, use the DESCRIBE command to see which items are key items and therefore can be tallied.

The EACH option displays each distinct value and its count. This example shows a tally of ETHNIC ORIGIN (C8), which displays all distinct values and counts for C8. The EACH option is in effect by default.

```
TALLY ETHNIC ORIGIN:
*******************************************
 ITEM-          ETHNIC ORIGIN
*******************************************
OCCURRENCES     VALUE
          2     AMERICAN INDIAN
          2     ASIAN
          7     BLACK
         31     CAUCASIAN
          8     HISPANIC

-------------------------------------------
          5 DISTINCT VALUES
         50 TOTAL OCCURRENCES
-------------------------------------------
```

If you choose the ALL option, only the lowest and the highest item values are displayed. The total number of distinct values always appears, along with the total number of occurrences for all the item's values. The ALL option gives you the totals quickly, without displaying each value.

This example is the same as the previous one, except ALL displays only the minimum and maximum values and totals.

```
TALLY /ALL/ ETHNIC ORIGIN:
*****************************************
 ITEM-   .      ETHNIC ORIGIN
*****************************************
 MINIMUM-       AMERICAN INDIAN
_________________________________________
 MAXIMUM-       HISPANIC
_________________________________________
        5 DISTINCT VALUES
       50 TOTAL OCCURRENCES
_________________________________________
```

Sometimes the LIMIT command is useful with the TALLY command. For example,

```
LIMIT 2, 0:
TALLY LAST NAME:
```

yields a tally of last names that occur two or more times. (The 0 means infinite.) To turn off all limits for subsequent commands, give the **END LIMIT command**. For details about the LIMIT command, see Chapter 3.8.

This last example shows how the limits of a minimum of 7 and a maximum of 28 occurrences affect the TALLY output.

```
LIMIT 7, 28:
TALLY/EACH/ETHNIC ORIGIN:
*****************************************
 ITEM-          ETHNIC ORIGIN
*****************************************
OCCURRENCES     VALUE
        7       BLACK
        8       HISPANIC
_________________________________________
        2 DISTINCT VALUES
       15 TOTAL OCCURRENCES
_________________________________________
```

TALLY is also a time-saver in checking for typos and accuracy of key values. For example, if a tally of LAST NAME yields 125 JONES values and one JOMES value, it could indicate an error in that last name.

Chapter 3.3: Printing Data Values

The **PRINT** (or **PR**) **command** displays the requested values, one line for each item's value. The PRINT command produces "quick and dirty" reports on a small volume of data. You can choose a subset of the data base by including a where-clause. If you do not give a where-clause, the entire data base is used. The components given after the keyword PRINT determine which items and records are displayed.

Here are some simple forms of the PRINT command.

PRINT ENTRY:
PRINT ENTRY where-clause:
PRINT component, component, ... :
PRINT /option, option, ... / component, component, ... :
PRINT component, component, ... where-clause:
PRINT component, component, ... , ordering-clause where-clause:

Many optional keywords and formatting choices are available. Some of the more frequently used options are briefly discussed below. They are illustrated at the end of this chapter. Here is a summary of the format options.

Defaults	Alternatives
SINGLE SPACE	DOUBLE SPACE
NUMBER	NAME
STUB	STUB SUPPRESS
NULL	NULL SUPPRESS
ZERO	ZERO SUPPRESS
INDENT	BLOCK
REPEAT	REPEAT SUPPRESS
TREE	RECORD

ENTRY (or **CO**) means display an entire logical entry, for example, all data for an employee. You can also display specific items, functions, or records. In the examples that follow, notice the item label displayed with each output line, followed by the system separator (asterisk) and then the value.

Item C-numbers are displayed if the **NUMBER option** is in effect. Use the **NAME option** if you want item names to appear, that is, PRINT/NAME/ ... followed by the rest of the command. To suppress the item labels completely and just look at values, use the **STUB SUPPRESS option**.

Output lines are indented according to various levels of data records if the **INDENT option** is in effect. For a left-justified display, use the **BLOCK option**. The **RECORD option** displays the

specified record but not descendants. **TREE** displays the specified record and all descendant data records.

If an item has no value in a record, nothing appears for it in your output. To display "-NULL-" for missing values, specify the **NULL option**.

After you change a format setting, the new setting remains in effect until changed again. Several format options (separated by commas) can be set in a command, for example, PRINT /NAME, INDENT, NULL/.

The example below selects employee number 1265 and displays employee number, last name, first name, and all values about positions in the company. The default format options are in effect. Notice all records below the POSITION WITHIN COMPANY (C100) record are displayed automatically because TREE (default) is in effect.

```
PRINT EMPLOYEE NUMBER, LAST NAME, FORENAME, POSITION WITHIN COMPANY
    WHERE EMPLOYEE NUMBER EQ 1265:
 1*  1265
 2* SLYE
 3* LEONARD R.

   101* GENERAL MAINTENANCE
   102* ADMINISTRATION & FINANCE
   103* SQT
   104* SUPPORT
   105* 04/02/1979

     111*       $3.50
     112* HOURLY
     113* 04/02/1979
     114*      $55.73

       121* 05/31/1979
       122*   80.00
       123*   12.00
       124*     $343.00
       125*      $37.20
       126*     $288.27

       121* 04/30/1979
       122*   80.00
       123*     .00
       124*     $280.00
       125*        $.00
       126*     $263.20
```

Notice the indentation (INDENT option) that shows the record levels in the data tree. Record levels and data trees are discussed in Chapter 5.1.

The following example illustrates the NAME option, which displays item names instead of numbers. Notice that C-numbers are given in the command. The where-clause locates three employees with the last name of Smith.

```
PRINT /NAME/ C2,C3,C6,C411,C412,C413 WHERE C2 EQ SMITH:
---
 LAST NAME* SMITH
 FORENAME* JERRY LEE
 SOCIAL SECURITY NUMBER* 823-10-0951
     SCHOOL* ST. LOUIS UNIVERSITY
     DEGREE/CERTIFICATE* BA
     DATE COMPLETED* 06/02/1965
 LAST NAME* SMITH
 FORENAME* JERRY LEE
 SOCIAL SECURITY NUMBER* 823-10-0951
     SCHOOL* ST. LOUIS UNIVERSITY
     DEGREE/CERTIFICATE* MA
     DATE COMPLETED* 06/18/1974
 LAST NAME* SMITH
 FORENAME* JANET F.
 SOCIAL SECURITY NUMBER* 105-32-9011
     SCHOOL* MARYMOUNT COLLEGE
     DEGREE/CERTIFICATE* BA
     DATE COMPLETED* 08/21/1969
 LAST NAME* SMITH
 FORENAME* GARLAND P.
 SOCIAL SECURITY NUMBER* 397-80-8491
     SCHOOL* AUSTIN COMMUNITY COLLEGE
     DEGREE/CERTIFICATE* AA
     DATE COMPLETED* 06/08/1976
```

The last example shows the use of the BLOCK and STUB SUPPRESS options to left-justify all output lines and suppress item labels.

```
PR /BLOCK,STUB SUPPRESS/ EMPLOYEE NUMBER,C3,C2,POSITION TITLE
   WHERE MANAGER EQ JLH:
---
  1123
 LEOPOLD
 FREEMAN
 SR SYSTEMS PROGRAMMER
  1049
 SOPHIA
 FERNANDEZ
 STANDARDS & PROCEDURES ANALYST
```

Chapter 3.4: Listing Values in Columns

The **LIST** (or **LI**) **command** displays item values across the screen in columnar format, with the item name as a column heading over the item's values. You can use the LIST command to display many values in a compact format and to create tailored reports.

Many options are available with the LIST command, such as a title or footer, user-supplied column headings, and your own spacing between columns (instead of the three blanks provided by SYSTEM 2000 DBMS).

To try the LIST command, begin with the one of the simplest forms, using default column headings.

```
LIST item,item,item ... :
LIST item,item,item ... where-clause:
LIST item,item,item ... , ordering-clause where-clause:
```

Notice that these simple formats are identical to the PRINT command; only the LIST keyword is different. Use these forms until you are comfortable with the PRINT and LIST commands, ordering-clause, and where-clause.

To specify your own column headings and title text, use the **TITLE option**, which offers many variations. Some of the simpler options are discussed on the following pages. For more details see other SYSTEM 2000 manuals.

The following command lists employee numbers and names for employees in the Corporation department.

```
LIST EMPLOYEE NUMBER, LAST NAME
     WHERE DEPARTMENT EQ CORPORATION:
---
*EMPLOYEE NUMBER   LAST NAME
***
*            1001   WATERHOUSE
*            1003   SALAZAR
*            1002   BOWMAN
*            1101   NATHANIEL
*            1006   GARRETT
*            1004   KNIGHT
                .      .
                .      .
                .      .
```

To supply a title line for your listing, specify the TITLE option with the **D option**. The current date will be centered below your title.

In this example, D(7) means to start the title in column 7. The listing shows first names (C3) and last names (C2) of employees in the Marketing department. Column headings are item names because none were specified in the command.

```
LIST /TITLE D(7)MARKETING EMPLOYEES/ C3,C2
  WHERE DEPARTMENT EQ MARKETING:

        MARKETING EMPLOYEES
            01/15/1987

 * FORENAME              LAST NAME
 ***
 * OLAN M.               GARRETT
 * VIRGINA P.            BROWN
 * FANNIE                LITTLEJOHN
 * MADISON A.            SCHOLL
 * RUBEN R.              BROOKS
 * ARMANDO               JUAREZ
 * DAVID                 AMEER
 * TRAVIS Z.             RICHARDSON
 * MERRILEE D.           WAGGONNER
 * ALAN F.               GOODSON
 * JERRY LEE             SMITH
 * LELAND G.             SHROPSHIRE
 * TAI                   CHAN
 * GWENDOLYN             VAN HOTTEN
 * ROMUALDO R.           RODRIGUEZ
 * JANICE L.             WILLIAMSON
 * GEORGE J.             GIBSON
```

To give your own column headings, use the **L(w)** or **R(w) options**. The **w** indicates how wide the column should be. The **L** means left-justify the column heading over the columnar display of values. The **R** means right-justify the column heading. If you omit L and R, the heading is left-justified by default. Each column heading can be one to three lines; a plus sign specifies the beginning of a new line for that heading.

If you use your own column headings, you must specify a heading for each item in the order of the specified items to be displayed.

The word **TITLE** and the **slashes** must be given if you specify a report title line or column headings.

Here is an example of left- and right-justified column headings. Notice the plus sign that makes ETHNIC ORIGIN a two-line column heading; L(9) causes this column to be nine characters wide.

```
LIST /TITLE R(11)EMPLOYEE, L(10)NAME, L(9)ETHNIC+ORIGIN/
   FORENAME, LAST NAME, ETHNIC ORIGIN WHERE SEX EQ FEMALE:
---
 *    EMPLOYEE   NAME          ETHNIC
                               ORIGIN
 ***
 * MOLLY I.      PITTS         BLACK
 * YOLANDA       SALAZAR       HISPANIC
 * ALTHEA        KNIGHT        CAUCASIAN
 * VIRGINA P.    BROWN         BLACK
 * FANNIE        LITTLEJOHN    AMERICAN
                               INDIAN
 * MERRILEE D.   WAGGONNER     CAUCASIAN
 * GWENDOLYN     VAN HOTTEN    CAUCASIAN
 * JANICE L.     WILLIAMSON    CAUCASIAN
 * CARRIE ANN    FAULKNER      CAUCASIAN
 * SOPHIA        FERNANDEZ     HISPANIC
 * JANET F.      SMITH         CAUCASIAN
 * PATRICE R.    KNAPP         CAUCASIAN
 * PATSY         MUELLER       CAUCASIAN
 * PENNY         SCHMIDT       CAUCASIAN
 * LILLIAN       COLLINS       CAUCASIAN
 * RITA M.       JONES         HISPANIC
```

This last LIST example shows a combination of options available for producing nicely formatted LIST output. The command contains a title, user-specified column headings, and an ordering-clause. The where-clause selects May 1979 payroll data for the Information Systems department.

```
LIST /TITLE D(15)PAYROLL WORKSHEET FOR MAY 1979, EMPLOYEE NAME,
L(11)SOC SEC NO,BASE HRS,OVERTIME+HRS,R(9)GROSS/C2,C6,C122,C123,C124,
OB C2 WH C102 EQ INFORMATION SYSTEMS AND PAYROLL MONTH EQ 05/31/1979:
---
```

PAYROLL WORKSHEET FOR MAY 1979
01/20/1987

* EMPLOYEE NAME	SOC SEC NO	BASE HRS	OVERTIME HRS	GROSS
***				
* CAHILL	102-78-8765	184.00		$2,700.00
* FERNANDEZ	764-91-0193	184.00		$2,200.00
* FREEMAN	828-26-7282	184.00		$1,750.00
* GARCIA	678-23-0123	184.00		$1,200.00
* HERNANDEZ	123-12-0987	184.00		$2,600.00
* JOHNSON	321-32-9446	184.00		$1,200.00
* JONES	543-87-1934	184.00		$1,500.00
* PITTS	462-01-0234	184.00	12.00	$1,400.00
* POLANSKI	497-36-7845	184.00		$2,600.00
* QUINTERO	339-94-2674	184.00		$2,000.00
* REDFOX	210-65-2786	184.00		$1,800.00
* REED	875-15-1388	184.00		$775.00
* REID	441-04-0121	184.00	10.00	$1,300.00
* SAVAGE	211-95-9608	184.00		$900.00
* SEATON	286-04-6279	184.00		$900.00
* SMITH	105-32-9011	184.00		$1,500.00

Much ad hoc information can be gathered and displayed with a simple LIST command. For more complex listings you might want to use the Genius feature, which prompts you conversationally for headings, titles, and so on. Or you can use the QueX software to browse through your own specific view of the data records, one by one, selectively.

Chapter 3.5: Selecting the Data You Want

A where-clause in a retrieval command selects specific records to be displayed. If a command has no where-clause, the system scans the entire data base. The where-clause also selects the records or values to be modified in update commands.

Each where-clause begins with **WHERE** (or **WH**). Where-clauses have many options, but here are some of the simplest formats.

```
WHERE condition
WHERE condition AND condition ...
WH condition OR condition ...
WH (condition AND condition) OR condition ...
```

Each **condition** gives some type of criterion for testing the values of an item. The format of a condition is one of the following.

item EXISTS	finds all existing values for an item.
item FAILS	finds missing values (nulls) for an item.
item operator value	compares item values with a given value.
item SPANS value*value	finds a range of values.

The value is any string of characters conforming to the type of the specified item. For example, for an INTEGER item, the value must be a whole number. The operator is one of the following:

EQ	equals	**NE**	not equals
GT	greater than	**LT**	less than
GE	greater than or equals	**LE**	less than or equals

CONTAINS contains the value as a word, suffix, or prefix.

Here are a few examples of where-clauses having only one condition.

```
... WHERE LAST NAME EQ GIBSON:
... WHERE C1 GT 1020:
... WH ZIP CODE SPANS 78660 * 78731:
... WHERE LAST NAME CONTAINS SON:
... WHERE COMMENT TEXT CONTAINS THE WORD PROGRAMS:
... WHERE ZIP CODE FAILS:
```

Most simple where-clauses have only one condition. However, two or more conditions in the same where-clause is not unusual. To specify more than one condition, use the **AND** and **OR operators**.

AND means both conditions must be true.
OR means either condition or both conditions must be true.

As in other parts of SYSTEM 2000 commands, **item** can be a C-number or an item name. Here are a few sample where-clauses with AND and OR operators.

```
... WHERE DEPARTMENT EQ MARKETING AND SEX EQ MALE:
... WH PAY RATE LE 800.00 AND PAYROLL MONTH GT 04/01/1979:
... WHERE C8 NE CAUCASIAN AND C7 EQ FEMALE:
... WH SKILL TYPE EQ COBOL OR SKILL TYPE EQ FORTRAN:
... WH C1 SPANS 1020*1051 AND DEPARTMENT EQ MARKETING:
... WHERE START DATE EXISTS AND C102 NE MARKETING:
... WHERE MAJOR FIELD EQ MATH OR MINOR FIELD EQ MATH:
```

When AND and OR are used in a single where-clause, the ANDs are processed first, then the ORs. If you need to alter the processing order, you can use pairs of parentheses to cause the enclosed sets of conditions to be processed first.

Other options, such as **AT**, **NOT**, **AND NOT**, **OR NOT**, and **HAS**, are available for more complex selection by position and by the non-existence of records or values. These options are discussed fully in other SYSTEM 2000 manuals.

You must use the **HAS operator** if you use the AND operator to search for two conditions on the same item. For example, suppose you want to see which employees know both COBOL and PASCAL. If you requested SKILL TYPE EQ COBOL AND SKILL TYPE EQ PASCAL, you would not select any records. The skill type is either COBOL or PASCAL but not both values in the same record. To select the correct employees, use the HAS operator with a parent record type above the level of JOB SKILLS. For example, the command

```
PR EMPLOYEE NUMBER WH ENTRY HAS SKILL TYPE EQ COBOL AND
     ENTRY HAS SKILL TYPE EQ PASCAL:
```

selects the logical entries that have both skills somewhere in their data records. The HAS operator has many uses and can be applied at all levels of a data structure.

The **SAME operator** duplicates a previous where-clause. SAME saves you time since you don't have to enter the same where-clause again. It also saves processing time, because the system saves the list of records that were qualified for the previous where-clause. The effect is dynamic (cumulative) unless you specify **SAME IS STATIC**, which saves the previous where-clause until you specify another where-clause not containing SAME.

Normally, you will want to use the SAME operator dynamically as illustrated below.

```
PR ... WHERE DEPARTMENT EQ MARKETING:
PR ... WHERE SAME AND SEX EQ MALE:
LIST ... WHERE SAME AND HIRE DATE GT 12/31/1976:
```

The first command selects all employees in the Marketing department. The second command selects all male employees in the Marketing department. The third command selects all male employees in the Marketing department who were hired after 1976.

If you place the SAME IS STATIC command after the first where-clause, the results are different.

```
PR ... WHERE DEPARTMENT EQ MARKETING:
SAME IS STATIC:
PR ... WHERE SAME AND SEX EQ MALE:
LIST ... WHERE SAME AND HIRE DATE GT 12/31/1976:
```

The first command selects all employees in the Marketing department. The second command freezes the "same" results until the next where-clause that does not contain a SAME operator. The third command chooses the male employees in the Marketing department. The last command chooses all employees in the Marketing department who were hired after 1976 -- not just the male employees in the Marketing department hired after 1976.

Chapter 3.6: Sorting Your Output

The **ordering-clause** sorts retrieval output if a where-clause is present in the command. If there is no where-clause, the output is in logical order, that is, according to the data structure of the records.

The ordering-clause begins with **ORDERED BY** (or **OB**), followed by the sort items, separated by commas. The comma before the ordering-clause is required. There is no comma before the word WHERE.

PRINT ... ,ORDERED BY sort key, sort key, ... where-clause:

LIST ... , ORDERED BY sort key, sort key, ... where-clause:

A sort key can be a key item, a non-key item, or a function. The order of sorting is major-to-minor from the first to the last specified sort key. The first sort key is the "major" item on which the data will be sequenced; the last sort key is the "minor" key. For example, if you want your data sorted by social security number within last name within department, you would specify

... ORDERED BY DEPARTMENT, LAST NAME, SOCIAL SECURITY NUMBER

Optionally, each sort key can begin with the keyword **HIGH** or **LOW**, with LOW being the default. LOW means to sort the item values in ascending order; HIGH means descending order.

For example, to print employee numbers (C1) and last names (C2) for all female employees in ascending sequence by employee number, give this command.

PRINT C1, C2, ORDERED BY C1 WHERE SEX EQ FEMALE:

The next example lists the employee numbers for those employees who know COBOL, in order by descending date of hire. Notice that the sort key HIRE DATE does not have to be included in the list of items to be displayed.

LIST C1, OB HIGH HIRE DATE WHERE SKILL TYPE EQ COBOL:

The last example has two sort keys. It displays employee numbers and names for all employees except those in the Information Systems department, ordered by department (C102) and by position titles (C101) within each department.

PR C1, C2, OB C102, C101 WHERE C102 NE INFORMATION SYSTEMS:

Chapter 3.7: Repeating a Command

SYSTEM 2000 DBMS offers many options and conveniences in the Self-Contained Facility. The SAME option was illustrated in the where-clause discussion.

The **DITTO** (or **DI**) **command** repeats everything on the left side of the where-clause, including the ordering-clause, if any. DITTO saves time when you want to perform the same retrieval or update with different selection criteria. The commands below display employee numbers, last names, and first names sorted by employee number for the Marketing department and then for the Finance department.

```
PRINT C1, C2, C3, OB C1 WH DEPARTMENT EQ MARKETING:
DITTO WH DEPARTMENT CONTAINS FINANCE:
```

Chapter 3.8: Limiting Your Output

The **LIMIT command** places a limit on the number of records selected for retrieval or updating. If the where-clause selects too few or too many records, the system issues a message and ignores the command. LIMIT has many options, but these are the most frequently used formats.

LIMIT number:	specifies the exact number of records
LIMIT min, max:	specifies the minimum and maximum limits
END LIMIT:	turns off all limits

Number, **min**, and **max** must be positive numbers or zero. Zero means infinite. For example, LIMIT 4,0 means no less than 4 and an infinite maximum. Limits stay in effect until you issue an **END LIMIT command**.

The LIMIT command controls the number of records selected for printing or updating. If you accidentally form a where-clause that selects more records than you intended, the request can take a long time to process; also, it would be tedious to scan the volume of records displayed on your screen. Use the COUNT function or the TALLY command to see how many records you can expect to select. Then set your limits accordingly before requesting massive retrievals or updates.

The LIMIT command is especially effective when removing data or records. For example, to ensure that only one record is selected, give **LIMIT 1** before the REMOVE command. If more than one record is chosen, the command is not processed; you can try again with new selection criteria.

PART 4: UPDATING DATA

The QUEST language offers two basic types of update commands: value modification and structure modification. The first type modifies values within selected records but does not delete or insert records. The second type of updates modifies the data tree structures as well as the values. A **data tree** is a data record and all of its related records at subordinate levels.

Here are the most commonly used **value modification commands.**

ADD adds values where none currently exist.

CHANGE changes existing values.

REMOVE deletes values.

Here are the most commonly used **structure modification commands.**

INSERT TREE inserts a new data tree of records and values.

REMOVE TREE deletes a data tree of records and values.

The scope of updating depends on the selection criteria in your where-clause. You can modify specific values in selected records. Or, you can perform global changes across the data base with a single command.

Records can also be located according to their position, for example, the first one or the last one. Trace notation specifies the record's position. Most updates do not require a trace because you can easily locate the records to be modified with a where-clause. See other SYSTEM 2000 manuals for more details about traces.

The output from an update command is either an informative message giving the number of selected records or a message saying that the data base was unaltered, which means that no updates were made. (The system will not modify a value if you try to change it to its current value.)

You might also want to experiment with the **ASSIGN command** and the **ASSIGN TREE command.** These commands assign values and replace record structures regardless of the existing contents in the data base. Use them carefully.

Chapter 4.1: Adding Values to Existing Records

The **ADD** (or **AD**) **command** adds a value for an item where no value currently exists. That is, it changes a null item in one or more records to a valued item. The simplest forms of the ADD command are shown below.

ADD item EQ value * where-clause:

ADD record EQ item*value* item*value* ... END* where-clause:

Notice the first format contains a single item name (or C-number). The second one specifies a record, which lets you add values for more than one item a designated record. If you specify a record, all items to be added must belong to that record. However, the item*value* pairs can be given in any order. The item specification in the item*value* pair must be the item number without the letter C.

In these examples, notice the required system separator (asterisk by default) after each value and between the item number and its value.

ADD SECURITY CLEARANCE EQ 224* WHERE C1 EQ 1265:

ADD C410 EQ 414* ENGLISH* 415* HISTORY* END* WH C1 EQ 1265:

The easiest way to find missing values for items is to set the NULL option in a PRINT or LIST command. This option causes -NULL- to be displayed where a value is missing in a record. Also, since the TALLY command displays values in sorted order, a visual check of TALLY output can identify missing values.

You can also isolate missing values with the FAILS operator in a where-clause condition. For example,

PRINT EMPLOYEE NUMBER, LAST NAME, FORENAME,
OB EMPLOYEE NUMBER WHERE SOCIAL SECURITY NUMBER FAILS:

gives you a list of employees whose social security numbers are missing. Then you could give a series of ADD commands, one for each listed employee. For example,

ADD SOCIAL SECURITY NUMBER EQ 111-22-3333* WH C1 EQ 4518:

adds 111-22-3333 as the social security number for the employee with the employee number of 4518. You could store this command as a parametric string if you will be using it frequently. (See Part 6.)

Chapter 4.2: Changing Values

The **CHANGE** (or **CH**) **command** changes existing values to different values.

The simplest forms of the CHANGE command are shown below.

CHANGE item EQ value * where-clause:

CHANGE record EQ item*value* item*value* ... END* where-clause:

Notice that these formats are identical to those used for the ADD command. The first format contains a single item name (or C-number). The second one specifies a record, which lets you change values for more than one item in the designated record type. If you give a record C-number or name, all items must belong to that record type. However, the item*value* pairs can be given in any order. The item specification in the item*value* pair must be the item number without the letter C. You don't have to give item*value* pairs for all items in a record; specify only the items whose values need to be changed.

A system separator (the asterisk by default) is required after each value and also between the item number and its value for the second format.

With the first format you can change the value for one item in a specific record or an item's value in many records, depending upon the selection criteria you designate in the where-clause.

You can easily use the CHANGE command to correct typographical errors in data base values. Also, the CHANGE command is useful in standardizing your data. For example, if the data base contains items that require specific keywords, use the TALLY command first. If the TALLY output shows ELECTRIC and ELECTRICAL, you could standardize every instance of these values to ELECTRIC with

CH KEYWORD EQ ELECTRIC* WHERE KEYWORD EQ ELECTRICAL:

The CHANGE command lets you keep your data accurate and current. You can load volumes of data, then standardize and correct the values later as necessary.

Suppose Molly Gibson, with employee number 1043, moves from the Information Systems department to the Documentation department. Change the DEPARTMENT value for her with the following command.

CH DEPARTMENT EQ DOCUMENTATION* WH C1 EQ 1043:

Chapter 4.3: Removing Values

The **REMOVE** (or **RE**) **command** has a variety of options. The most common use is for removing the values for one item in the records selected by the where-clause. Removing means changing item values to nulls. The actual records in the data tree structure are never removed except by the REMOVE TREE command, discussed in Chapter 4.5.

The most frequently used format for the REMOVE command is

REMOVE item where-clause:

For example, to remove a March 1, 1978 deduction for employee number 1120, enter

REMOVE CURRENT DEDUCTION WH C113 EQ 03/01/78 AND C1 EQ 1120:

To remove all values of minor field (C415) for all employees, enter

REMOVE MINOR FIELD WHERE MINOR FIELD EXISTS:

Chapter 4.4: Inserting New Records

The **INSERT TREE** (or **IT**) **command** is handy for loading small amounts of new data. Use it when setting up logical entries in a model data base. (For loading larger volumes of new data, there are other options. For example, you might be able to use the QueX feature, as discussed in Chapter 7.3. See Chapter 5.3 for other alternatives, such as PLEX programs, to load machine-readable source data and volume production data that needs editing or validation.)

A simple form of the INSERT TREE command inserts a new logical entry as the last logical entry, indicated by the *0 trace notation.

INSERT TREE ENTRY *0 EQ value stream ... END*:

The value stream consists of a list of item*value* pairs. *END follows the value stream. If an item's value is not available, don't include the item number in the value stream. For example,

IT ENTRY *0 EQ 1*2199* 2*JOHNSON* 3*MARY* END*:

loads a new logical entry for employee number 2199, Mary Johnson. Items 1, 2, and 3 are valued in the record after this command is processed. Other items in the record are null. No data records are below this level 0 record.

To insert descendant records for a new logical entry, use the following format, where record* is a descendant record number.

IT ENTRY *0 EQ value stream record* value stream
record* value stream ... END*:

For example, to insert a new logical entry for Jane Simpson, including some descendant records for her position within the company (C100) and salary data, you could enter the command shown below.

Notice record number 100* followed by the item*value* pairs for items 101 and 102, then record number 110 followed by the item*value* pairs for items 111 and 112. Parent record values must be given before the descendant record values. Here, the trace (*1) inserts the new logical entry as the first one.

IT ENTRY *1 EQ 1*2199*2*SIMPSON*3*JANE*100*101*SECRETARY*102*
MARKETING*110*111*900.0*112*MONTHLY*END*:

INSERT TREE can also be used to insert individual records or subtrees into existing data trees.

Chapter 4.5: Removing Records

The **REMOVE TREE** (or **RT**) **command** removes data records and their values from the data base. REMOVE TREE affects the structure of data trees. It deletes each selected record and all its descendant records, which means that entire logical entries or subtrees within logical entries can be removed.

Here are the simplest forms of the REMOVE TREE command.

REMOVE TREE ENTRY where-clause:

REMOVE TREE record where-clause:

The first format removes entire logical entries. The where-clause specifies which ones are to be removed. For example,

REMOVE TREE ENTRY WHERE C1 EQ 1032:

removes the level 0 record and all descendant records for the employee whose employee number is 1032. No data would be left in the data base for this employee.

The second format removes one or more subtrees. Here, **record** is the C-number or name of the record at the top of the subtree to be removed. For example, the command

RT C100 WH C1 EQ 1120 AND C101 EQ ASSISTANT PROGRAMMER:

affects the data records for employee number 1120. It removes the POSITION WITHIN COMPANY (C100) record that contains POSITION TITLE (C101) equal to Assistant Programmer. It also removes all descendant records, such as SALARY WITHIN POSITION (C110), that exist below the selected C100 data record for this employee.

Sometimes it might be more convenient to select the records to be removed by their position in the data base, for example, the last or the first or the third logical entry. Use a trace to isolate data records by their position. The simplest form of trace notation is that of isolating a logical entry. In the examples below, 0 means the last logical entry and 2 means the second logical entry.

REMOVE TREE ENTRY *0:
RT ENTRY *2:

A trace can be very simple or very complex, depending on the level of the records affected.

PART 5: DEFINING AND LOADING A DATA BASE

Part 5 discusses how to define a new data base using the DEFINE language and various methods for loading your data. SYSTEM 2000 terminology is important for understanding this material. Chapter 5.1 defines some basic terms. See the SYSTEM 2000 Glossary and the DEFINE Language manual for more terms and detailed definitions.

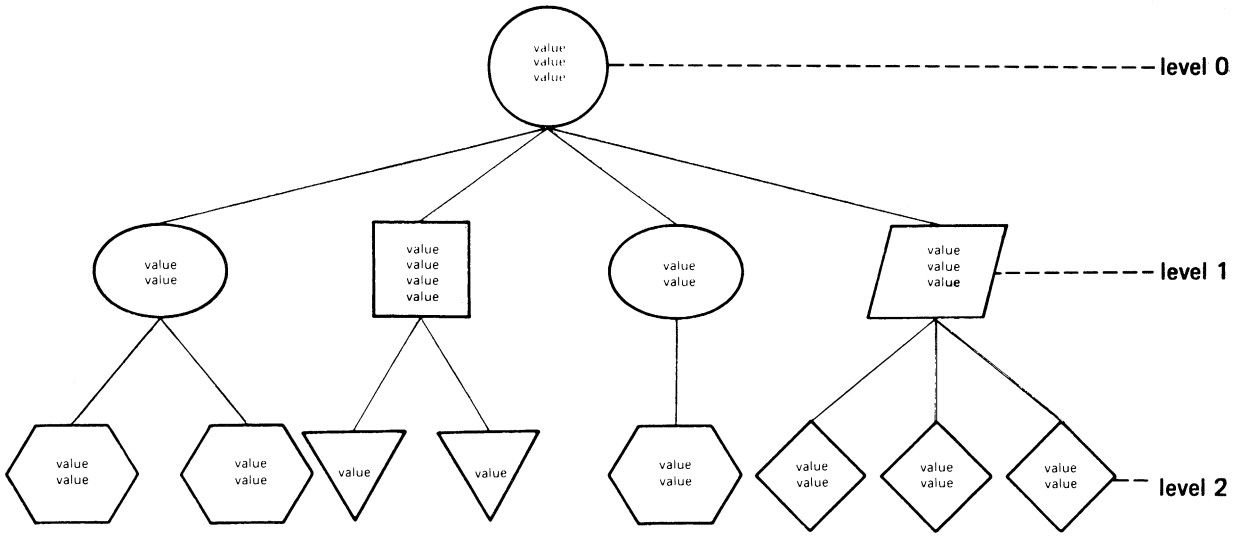
Chapter 5.1: Terminology

A **data base** is an organized collection of interrelated data that serves one or more applications. The user specifies the types of data and the record relationships for the data base with the DEFINE language. These specifications become the **data base definition**. The definition is used by both the user and SYSTEM 2000 DBMS for storing, modifying, and retrieving information.

A SYSTEM 2000 data base consists of groups of data values called **data records** organized in a hierarchical structure so that some records are subordinate to others. Each record can contain one or more **items** that are used to store the actual data values.

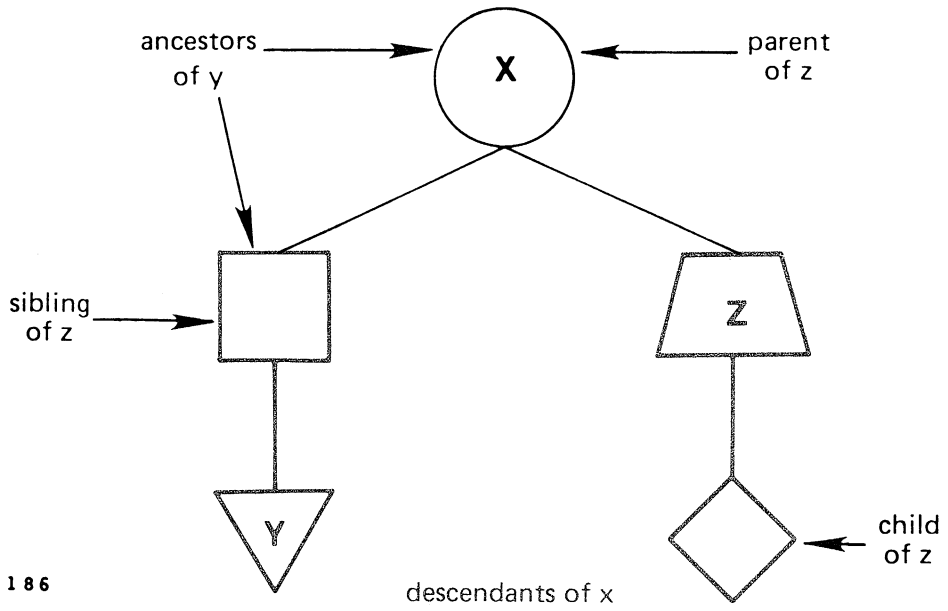
A data base definition can also include **strings** and **functions**. These are not part of the hierarchical structure of a data base and do not have data values. A **string** lets you store frequently used commands or parts of commands. A **function** lets you store an arithmetic calculation. Like items and records, you give each string and function a C-number and a name. This provides an easy way to call them during a session, saving you the time of formulating the commands and calculations every time you use them.

A SYSTEM 2000 data base is a collection of **logical entries** (or **entries**), each entry having one or more data records. The records are related logically into a **data tree**, with only one record at level 0 for each logical entry and, usually, one or more **subtrees** extending down through level 1, level 2, and so on.



1185

In any tree, certain relationships always exist among the records. The terms **parent**, **child**, **ancestor**, **descendant**, and **sibling** represent the same relationships in a tree as they would if the records were the names of persons in a family tree. The difference is that, in a data base, every record has only one parent, and except for the level 0 records, nobody is an orphan.

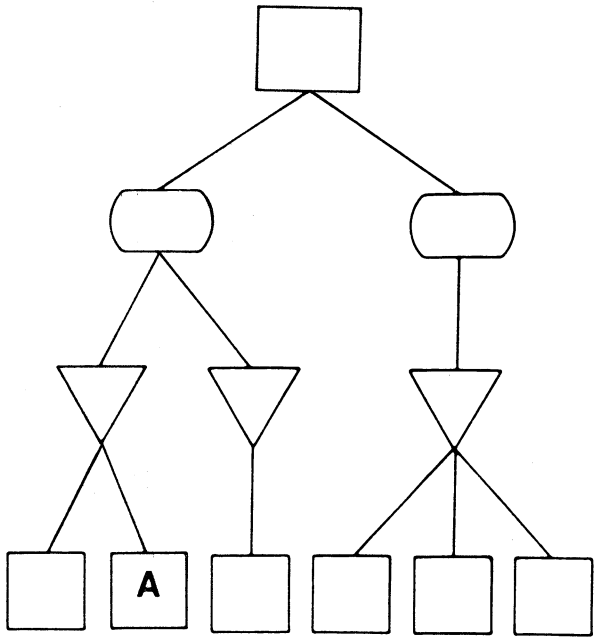


1186

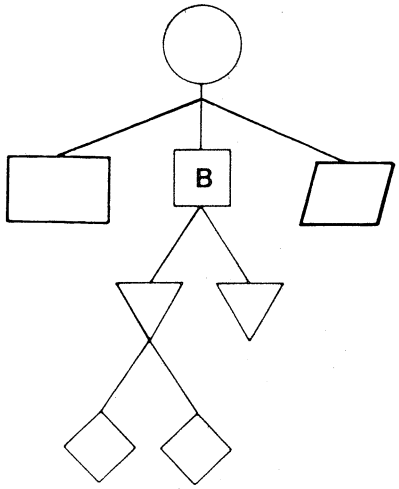
A few other terms commonly used with data trees are **level, path,** and **family.** A **level** in a data tree or in the definition resembles a generation of people. The level number of a record equals the number of ancestors that the record has.

The **path** of a record is the set of records consisting of the record and all its ancestors. The **family** of a record consists of the record, all its ancestors, and all its descendants.

Path of A



Family of B



1325

The path of record A is A and its three ancestor records. Record A is at level 3 since it has three ancestors.

The family of record B is B, its parent, and all of its descendant records. Record B is at level 1 since it has only one ancestor.

Chapter 5.2: Defining a Data Base

Before you actually define a new data base, jot down the various kinds of data you want to store. For example, in a MAILING LIST data base, you will want the name and address of each person. You might also want to store the subscription expiration date if the list is for magazines. Does the person want the mail sent to a business address or to a home address? Do you want to store a person's name once and have specific records at level 1, each containing the name of a magazine he is to receive?

Are the values alphanumeric, integer, decimal, or date? Will you use the item frequently as selection criterion?

Next, make a rough outline of which items belong together in the different records. Start small and use SYSTEM 2000 DBMS to help you analyze your requirements.

Define a simple data base using the DEFINE language. (You might also be able to use the CREATE facility if it is available at your site.) Enter some data with INSERT TREE commands. Try QUEST commands to describe, print, list, tally, and update your data. Get the "feel" of SYSTEM 2000 DBMS and your data and your application needs while in this modeling stage. In very little time, you will become familiar with the basic capabilities.

As you begin to use prototype definitions, you will learn what modifications might be necessary. You can add more items to a record or add more lower level records. Store some data; change the model; try again. Through experimenting, your real data base will evolve and grow.

Section 5.2.1: DEFINE Sessions

To define a new data base, call SYSTEM 2000 DBMS and give a password that will become the master password for the new data base. Then specify the name of the new data base with the **NEW DATA BASE command** (abbreviated NDB). See Chapter 2.3 if you have forgotten how to do this. Begin entering the item and record definitions and, when finished, enter the MAP command.

SYSTEM 2000 DBMS checks your definitions as you enter them to see if there are any inconsistencies. When you give the **MAP command**, all your definitions are stored in the new data base. At this point, no data values or records exist, but item attributes and record relationships are all defined.

You can call the DEFINE language any time to modify the definition. Always use the DESCRIBE command after any change to see your modified definition, so you can make sure it is correct before loading volumes of data. Try storing a few small logical entries, enough to test some PRINT or LIST commands.

The following commands illustrate a DEFINE session that creates a sample mailing list data base. Rules for specifying record and item definitions are discussed next.

```
USER,ME:
NDB IS MAILIST:
ECHO ON:
1* ID (INTEGER 9(6)):
2* NAME (CHAR X(30)):
3* STREET ADDRESS (NON-KEY CHAR X(45)):
4* CITY (NON-KEY CHAR X(15)):
5* STATE (CHAR XX):
6* ZIPCODE (INTEGER 9(5)):
10* LAST UPDATE (DATE):
MAP:
```

At first glance this definition seems very short. However, it is a valid definition and could be used to store thousands of mailing addresses.

Notice the user-assigned numbers and names for the items. All items are at level 0, which means the data base will consist of level 0 records only.

The attributes for each item are given in parentheses, for example, the type of values to be stored and number of characters expected. CHAR means alphanumeric characters, whereas INTEGER means whole numbers. CHAR X(30) means 30 characters. NON-KEY means you don't plan to use the item in where-clauses very often. KEY is the default. These attributes are explained in more detail next.

Section 5.2.2: Defining Items and Records

Four types of components can be defined for a data base: **items**, **records**, **strings**, and **functions**. This discussion presents some of the simplest forms of item and record definitions. String and function definitions are discussed in Part 6. For more information, see the DEFINE Language manual.

The format of each component definition is

component number * component name (component attributes):

Component numbers can have a maximum of 4 digits in the range of 1 through 4095. Each **component name** can have up to 250 characters. Certain reserved words, such as WHERE, cannot be used. SYSTEM 2000 DBMS automatically sets the labels for the level 0 record to C0 and ENTRY. All other numbers and names are user-specified. You can use either the name or the number in command syntax.

Item attributes include the item type, the anticipated length of the data values (picture), whether the values are to be indexed for faster access (KEY/NON-KEY), and the association of items to records (IN). Only the item type is required. SYSTEM 2000 DBMS provides defaults for the other attributes.

item definitions

The two general formats for defining item attributes are shown here. If the word KEY is omitted, the item is a key item by default.

(KEY item type picture IN record padding)

(NON-KEY item type picture IN record padding)

Before discussing each of these attributes, here are a few examples to provide a context.

```
(CHAR X(20))
(DATE)
(INTEGER 9(6) IN 210)
(NON-KEY DECIMAL 999.9 IN 100)
(KEY MONEY $9999.99 IN 325 WITH MANY FUTURE ADDITIONS)
```

For more examples of component definitions, see the DESCRIBE output in the fold-out.

item type

The most commonly used item types are

CHAR	for alphanumeric values
DATE	for dates
INTEGER	for whole numbers
DECIMAL	for numbers with a decimal point
MONEY	for decimal numbers with a dollar sign.

picture (length of values)

The length of the anticipated values is known as the picture. The default pictures shown below are set automatically if you don't specify a picture for the item.

CHAR defaults to 7 characters; the maximum is 250. You should choose a size that will hold most of the values for the item. However, if some values are longer, the system automatically stores the extra characters in an overflow area. To specify your own picture, use the notation X(n). For example, x(40) allows 40 characters before overflowing.

DATE does not need a picture specification because the size is constant and is set by the system.

INTEGER defaults to 7 digits; the maximum is 15. No overflow is allowed. To give your own picture, use the notation 9(n). For example, 9(4) allows 1 to 4 digits.

DECIMAL defaults to 6 digits on the left of the decimal point and 2 digits on the right. You can have a total of 15 digits with up to 10 digits on one side of the decimal point. No overflow is allowed. To specify your own picture, use the notation 9(m).9(n). For example, 9(3).9(4) indicate 3 digits on the left of the decimal point and 4 digits on the right.

MONEY is the same as type DECIMAL, but it includes a dollar sign and an optional CR for the minus sign.

KEY/NON-KEY

The KEY/NON-KEY specification indicates whether you want the values indexed for faster access. By default, all items are indexed (KEY) unless you designate NON-KEY when you define the item.

Indexes are real time-savers for selecting records. However, they do occupy space in the files, and they require processing time for updates. Therefore, for a large data base, you would probably want to define some items as non-key. Non-key items in a where-clause require a sequential search of all records.

If you find later that you have made the wrong indexing choice, a CONTROL language command is available to create or remove indexes, even after volumes of data have been stored.

IN record

The IN specification means an item belongs in a given record. Item definitions that do not contain IN are automatically associated with the level 0 record.

Level 0 items usually contain data that is unique for each logical entry. For example, each employee has only one birthday. Records at level 1, level 2, and so on, hold data that occurs more than once. For example, each employee can have several skills; therefore, several data records can be created, one for each skill.

To specify that an item belongs to a record, use IN followed by a record number. For example,

102* DEPARTMENT (CHAR IN 100):

means that the DEPARTMENT item (C102) is part of record C100.

padding

A padding option is available for key items. Padding reserves contiguous space for multiple occurrences of the same value. However, padding is not essential for defining a data base and can be added to the definition as the data base grows in a production environment. You specify padding with one of the following:

WITH FEW FUTURE OCCURRENCES
WITH SOME FUTURE OCCURRENCES
WITH MANY FUTURE OCCURRENCES

record definitions

To define the attribute for a record, use either of these formats.

(RECORD)
(RECORD IN record)

If there is no IN record specification, the record is automatically at level 1. That is, it becomes a child of the level 0 record.

If IN is used, followed by a record number, the record becomes a child of the specified record. For example, in the EMPLOYEE data base, the record definition

110*SALARY WITHIN POSITION (RECORD IN 100):

means salary records (C110) are children of POSITION WITHIN COMPANY (C100) records.

Section 5.2.3: Order of Definition

Although not required, the usual order for defining a data base is

```
all level 0 items
  a level 1 record (say, A)
    all items in record A
       a level 2 record subordinate to record A (say, B)
          all items in record B
            a level 3 record subordinate to record B (say, C)
               all items in record C
                        .
                        .
                        .
       another level 2 record subordinate to record A
                        .
                        .
                        .
  another level 1 record
              .
              .
              .
strings                 Strings and functions can be defined
functions               at any time; see Part 6.
```

The example below is the previous MAILIST definition with additional records at level 1 and level 2. The record for complaints is at level 1. The subordinate COMPLAINT SUMMARY record at level 2 stores comments describing the complaint. The SUBSCRIPTION record at level 1 allows each person to have more than one subscription, with the each magazine name and expiration date in a separate data record.

```
USER,ME: NDB IS MAILIST: ECHO ON:
1* ID (INTEGER 9(6)):
2* NAME (CHAR X(30)):
3* STREET ADDRESS (NON-KEY CHAR X(45)):
4* CITY (NON-KEY CHAR X(15)):
5* STATE (CHAR XX):
6* ZIPCODE (INTEGER 9(5)):
10* LAST UPDATE (NON-KEY DATE):
  100* COMPLAINT (RECORD):
       110* COMPLAINT DATE (DATE IN 100):
       120* COMPLAINT TYPE (CHAR X(5) IN 100):
       200* COMPLAINT SUMMARY (RECORD IN 100):
            210* COMPLAINT COMMENTS (CHAR X(30) IN 200):
  400* SUBSCRIPTION (RECORD):
       401* MAGAZINE (CHAR X(25) IN 400):
       402* EXPIRATION (DATE IN 400):
MAP:
```

Chapter 5.3: Loading the Data

SYSTEM 2000 DBMS offers several ways to load data. The method you choose depends on the amount of data, the available resources, whether new data is loaded periodically or needs to be added in an ad hoc fashion. Also, is the data base a model or a production data base?

You will probably use one of these two methods to load your first data bases.

INSERT TREE loads all records for a logical entry (for example, an employee). You can use it to create several model logical entries quickly. This command is described in Chapter 4.4.

QueX is a menu-driven facility. You can easily insert any number of logical entries, individual records, or subtrees. You navigate through the data base retrieving data or creating new data records by filling in labeled slots on the screen. Knowledge of programming and SYSTEM 2000 command language are not required. Chapter 7.3. gives more details about the QueX facility.

For loading large batches of data, other methods are available. After you are familiar with SYSTEM 2000 DBMS, you might want to investigate these alternatives.

The **LOAD command** loads many logical entries at once. You create an off-line file and edit it with a site text editor. The Data File thus created can then be read by the LOAD command in a Self-Contained Facility session.

A **PLEX program** can load large volumes of new data on a periodic basis. Also, the program can insert machine-readable data into a SYSTEM 2000 data base. PLEX requires knowledge of a higher level language, such as COBOL, FORTRAN, or PL/I.

The **CREATE facility** lets you define and load a new data base through a question/answer type of session. (Available only on specific versions of SYSTEM 2000 DBMS.)

PART 6: STORING FREQUENTLY USED COMMANDS

A data base definition can also include string and function definitions. These are not part of the hierarchical structure of a data base, and they do not have data values. A **string** lets you store frequently used commands or parts of commands. A **function** lets you store an arithmetic calculation.

Chapter 6.1: String and Function Definitions

A string can contain part of a command or one or more entire commands that are used frequently. A stored function can contain the same kinds of calculations as for the ad hoc functions discussed in Chapter 6.2.

Like items and records, you give each string and function a number and a name. Stored strings and functions are simple to call, and they save you from formulating commands and calculations every time you use them.

String and function definitions have these formats.

string number * string name (STRING $string contents$)

function number * function name (FUNCTION $function contents$)

Basically, strings are "pieces" of commands or whole commands, formatted exactly as though you entered them at your terminal. Chapter 6.3 contains an example of a string.

Here is an example of a simple stored function. For details about the types of functions and calculations that you can use, see Chapter 6.2.

2000*AGE WHEN HIRED (FUNCTION$((HIRE DATE-BIRTHDAY)/365.25)$):

To use this function in a PRINT command, for example, enter

PRINT LAST NAME, FORENAME, *C2000* WHERE SEX EQ MALE:

You can add, delete, and modify strings and functions any time. To look at them, give the DESCRIBE command.

DESCRIBE STRINGS:
DESCRIBE FUNCTIONS:

Chapter 6.2: Functions

Two types of functions are available for the PRINT and LIST commands: system functions and user-supplied functions.

System functions are keywords recognized as computations by the system.

SUM	sums numeric values.
AVG	averages numeric values.
MIN	finds the minimum value for an item.
MAX	finds the maximum value for an item.
COUNT	counts values or records.
SIGMA	obtains the standard deviation of numeric values.

A system function followed by an item or record label displays the results of that operation on the item or record. The following command counts the number of logical entries in the data base.

```
PRINT COUNT ENTRY:
---
 CNT 0*  51
```

To print the sum and average of vacation hours (C11) for all employees in the Information Systems department (C102), give this command.

```
PR SUM ACCRUED VACATION, AVG C11 WH C102 EQ INFORMATION SYSTEMS:
---
 SUM 11*  908.000
 AVG 11*  60.533
```

You can also use system functions within user-supplied functions.

User-supplied functions allow you to do arithmetic operations on item values, constants, or system functions. A user-supplied function must always be enclosed in parentheses. These functions have various formats. Some of the simple ones follow.

```
(component  operation  component)
(component  operation  constant)
(component  operation  component  operation  component ...)
(system function (user-supplied function))
```

The following symbols designate the type of arithmetic operation:

+	for addition
-	for subtraction
/	for division
*	for multiplication.

To list the total hours of accumulated vacation and sick leave for each corporate employee, enter this command.

```
LIST LAST NAME,(ACCRUED VACATION + ACCRUED SICK LEAVE)
     WHERE DEPARTMENT EQ CORPORATION:
 * LAST NAME
 ***
 * WATERHOUSE          16.000
 * SALAZAR            136.000
 * BOWMAN             120.000
 * NATHANIEL           48.000
 * GARRETT            140.000
 * KNIGHT              16.000
 * MILLSAP             72.000
 * FAULKNER            64.000
 * KNAPP               52.000
 * MUELLER             80.000
```

This command lists projected annual salaries sorted by gross pay.

```
LIST C2, (GROSS PAY * 12), ORDERED BY GROSS PAY
WHERE PAY SCHEDULE EQ MONTHLY AND PAYROLL MONTH EQ 05/31/1979:
 * LAST NAME
 ***
 * REED               9300.000
 * KNIGHT             9600.000
 * SAVAGE            10800.000
 * SEATON            10800.000
 * SMITH             11100.000
 * NATHANIEL         11400.000
 * JONES             12000.000
      .
      .
      .
```

You can access the current date in a calculation with the notation ***FTODAY***. This example computes the average years of employment for all employees.

```
PRINT AVG((*FTODAY* - HIRE DATE)/365.25):
---
 AVG 5001*  6.819
```

Chapter 6.3: Parametric Strings and Functions

Both strings and functions can be defined with variables in them, called **parameters**. With parameters, you can store the skeleton of a command or function. To define a parameter, use the number notation of the form *n*, for example, *1* or *2*. The numbers are consecutive. When you call the string or function, supply the actual values for the parameters in order, by position.

Here is a sample parametric string.

```
1775* NO. OF EMPLOYEES (STRING $PRINT COUNT ENTRY WHERE
      SEX EQ *1*:$):
```

This string has one parameter, the *1* in the where-clause. The value of male or female will be substituted for the *1* when you use the string.

To call a parametric string, enter an asterisk followed by the string's C-number or name and supply the required values in order by parameter number. Enclose the parameter values in parentheses. To call the string defined above, give either of these commands.

***C1775 (FEMALE)** counts the female employees

***NO. OF EMPLOYEES (MALE)** counts the male employees

Storing frequently used commands and functions saves time and avoids the chance of making errors each time you enter them. Also, specifying variable parameters makes them flexible. You can declare up to 32 parameters in one string or function.

Each string or function can have up to 250 characters. Also, you can nest stored strings and functions inside other strings and functions. Thus, very long command streams can be stored in the definition and processed with a call to one string that invokes other strings.

See the DEFINE language manual for more information about strings and functions.

PART 7: OTHER SYSTEM 2000® FEATURES

This part discusses other end-user and programmer facilities available with SYSTEM 2000 DBMS.

Chapter 7.1: Self-Contained Facility Options

This chapter discusses a few more capabilities available in the Self-Contained Facility, not described elsewhere in this manual.

Section 7.1.1: Calling the Various Languages

The Self-Contained Facility gives you access to these SYSTEM 2000 languages without leaving the session.

CONTROL assigns security, saves or restores data bases, and provides backup and recovery.

DEFINE defines the items, records, strings, and functions for the data base; it also modifies the data base definition.

QUEST retrieves and updates data.

REPORT composes and generates reports; it can be called from the QUEST language only.

QUEUE groups a set of retrievals or updates to be processed together; it can be called from the QUEST language only.

CREATE defines and loads a new data base through a question/answer session.

To call a language, enter the appropriate command from the list below. These are known as system-wide commands, because they are recognized by any of the Self-Contained Languages.

CONTROL:	calls the CONTROL language
DEFINE:	calls the DEFINE language
ACCESS:	calls the QUEST language
REPORT:	calls the REPORT language; must be in the QUEST language
QUEUE:	calls the QUEUE language; must be in the QUEST language
CREATE:	calls the CREATE facility

The QUEST and DEFINE languages have already been discussed in this manual. The REPORT language is discussed in Chapter 7.2.

Section 7.1.2: The CONTROL Language

CONTROL language commands let you save or restore a data base, initiate recovery procedures, log updates, and assign passwords with specific authorities.

The CONTROL language also contains commands for reloading a data base and compressing the internal data base tables. Except for saving data bases, CONTROL language commands are generally used by the data base administrator or the person in charge of controlling the use of the data base, that is, the master password holder.

The CONTROL language can be used during any Self-Contained Facility session. You can call it by simply giving the system-wide command **CONTROL:** whenever needed.

Section 7.1.3: The QUEUE Language

The QUEUE language is available through the QUEST language. It lets you place batches of retrievals and updates into a QUEUE session. Then the queued commands are processed as one group. QUEUE sessions differ from QUEST commands where each command is processed when encountered and the results return to the user immediately. Also, the QUEUE language syntax is slightly different than the QUEST language.

Section 7.1.4: The CREATE Facility

The CREATE facility is available for some versions of SYSTEM 2000 DBMS. CREATE is a conversational facility that lets you define a new data base, estimate its size, and load data.

Through a series of prompts and user responses, you can define the items and records for the new data base. Then you can collect and verify data, as well as validate fixed-field format data. Also, the CREATE facility helps you estimate the size of a new data base before actually loading the data.

Section 7.1.5: User Files

Four user files hold the input to SYSTEM 2000 DBMS and the output from SYSTEM 2000 DBMS.

Command File is an input file containing SYSTEM 2000 commands; set by default to your terminal.

Data File is an input file containing value streams of data; used to enter batches of new values into the data base; set by default to your terminal.

Report File is an output file containing results of retrieval commands; set by default to your terminal.

Message File is an output file containing messages from SYSTEM 2000 DBMS and echoes of your commands; set by default to your terminal.

Normally, the names of these files are transparent to you and of no concern while using SYSTEM 2000 DBMS. However, you might want to create a Command File and edit it outside a Self-Contained Facility session; then you can tell SYSTEM 2000 DBMS to process this file. Likewise, you can create, edit, and use a Data File created off-line.

Usually you will want all messages and retrieval results to come back to your terminal screen automatically. But there might be times when the results of a LIST or PRINT command need to be saved on an alternate Report File, for example, to be printed as hard copy later on.

System-wide commands are available for manipulating these user file names. See your data base administrator for how to set up these alternate, special user files.

Chapter 7.2: The Genius Facility and the REPORT Language

The Genius facility is a conversational tool for composing reports. Genius prompts you for appropriate responses throughout the session. Therefore, you do not need a thorough knowledge of SYSTEM 2000 DBMS or its REPORT language. Genius can be used by the novice and by the experienced data base user. Many defaults are provided.

The Genius facility automatically produces error-free REPORT language commands. You supply only a where-clause to select the required records.

If you want to compose your own reports (not through Genius), then you can call the REPORT language by entering either of the following commands from the QUEST language.

REPORT:

COMPOSE:

While Genius offers very complex report options and camera-ready listings, using the REPORT language directly gives some additional flexibility, such as logical paging or special edit, insertion, and replacement characters. Also, for very complex reports, you can use Genius to produce the REPORT language commands, then polish or modify those commands to produce special editing or spacing for the final results.

Here is a simple Genius session. The data is taken from a COMPANY data base. The example gives payroll information by employee. Results are ordered by last name (C2), and details appear for each payroll month (C121). Gross pay (C124) is listed, along with subtotals for each employee. Grand totals are at the end of the report.

```
                                GENIUS
                              RELEASE 2.0

*** PARAMETER CHANGE SECTION ***

    ENTER PARAMETER NAME AND NEW VALUE.  ENTER "LIST"
    TO DISPLAY.  ENTER <CR> WHEN FINISHED
LIST
      WIDTH             = 080
      LENGTH            = 055
      COUNT SIZE        = 06
      SUM INCREASE      = 2
      TITLES            = YES
      LINE SEPARATOR    = SPACE
      INPUT NAME        = INPUT
      RW DELIMITER      = "
      SYSTEM SEPARATOR  = *
      TERMINATOR        = :
      PROMPTS           = REGULAR
      PARAMETER CHANGE  = YES
      ECHO INPUT        = OFF
      TARGET CPU        = IBM
WIDTH 40
      WIDTH             = 040
LINE SEPARATOR -
      LINE SEPARATOR    = -


*** TITLE  SECTION ***

    ENTER ANY ITEM(S) THAT YOU WISH TO PRINT IN THE
    TITLE.  SPECIFY  "SAME PAGE" FOR ANY ITEM(S) THAT
    SHOULD NOT RESULT IN A NEW PAGE FOR EACH DISTINCT VALUE.
    ENTER <CR> IF NONE DESIRED.

    ENTER ITEM  1


    ENTER TITLE LINES TO BE PLACED AT THE TOP OF EACH PAGE
    (MAXIMUM OF 8 LINES).  ENTER <CR> WHEN FINISHED.

    ENTER TITLE LINE   1
EXAMPLE 1

    ENTER TITLE LINE   2
PAYROLL INFORMATION BY EMPLOYEE.

    ENTER TITLE LINE   3


     YOU HAVE ENTERED THE FOLLOWING:

  REPORT TITLES
    1) EXAMPLE 1
    2) PAYROLL INFORMATION BY EMPLOYEE.

     ENTER <CR> IF OK OR LINE NUMBER AND NEW VALUE
```

```
*** COLUMN CONTROL SECTION ***

    ENTER A VERB AND ITEM NUMBER FOR EACH COLUMN.
    THE FOLLOWING VERBS MAY BE USED IN A REPORT:
    BREAK, SORT
    COUNT, SUM, LIST, OR COMPUTE
    ALL BREAKS AND SORTS MUST PRECEDE ANY OTHER VERB.


- - - - - ENTER COLUMN 01 - - - - -
BREAK C2 13

    HEADING IS: LAST NAME
    ENTER <CR> TO ACCEPT OR ENTER NEW HEADING
—LAST NAME

    HEADING IS: --LAST NAME
    ENTER <CR> TO ACCEPT OR ENTER NEW HEADING



- - - - - ENTER COLUMN 02 - - - - -
SORT C121

    SORT ACKNOWLEDGED FOR: PAYROLL MONTH


- - - - - ENTER COLUMN 02 - - - - -
LIST C121

    ENTER "COUNT" IF COUNT OF LIST ITEM IS DESIRED
    ENTER <CR> IF NONE DESIRED


     HEADING IS: PAYROLL MONTH
     ENTER <CR> TO ACCEPT OR ENTER NEW HEADING
-PAYROLL- MONTH

     HEADING IS: -PAYROLL- MONTH
     ENTER <CR> TO ACCEPT OR ENTER NEW HEADING



- - - - - ENTER COLUMN 03 - - - - -
LIST C124

    ENTER "COUNT" IF COUNT OF LIST ITEM IS DESIRED
    ENTER "SUM" IF SUM OF LIST ITEM IS DESIRED
    ENTER <CR> IF NONE DESIRED
SUM 6.2

    THE HEADING MUST NOT EXCEED  16 PRINT POSITIONS.

    HEADING IS: GROSS PAY
    ENTER <CR> TO ACCEPT OR ENTER NEW HEADING
 GROSS PAY-  SUMMED-BY EMPLOYEE

    THERE ARE  4 PRINT POSITIONS REMAINING ON THE REPORT PAGE

    HEADING IS:  GROSS PAY-  SUMMED-BY EMPLOYEE
    ENTER <CR> TO ACCEPT OR ENTER NEW HEADING



 - - - - - ENTER COLUMN 04 - - - - -


    PLEASE STAND BY, GENIUS AT WORK.

    ENTER "END" TO TERMINATE SESSION OR <CR> TO PRODUCE
    ANOTHER LISTING.
END
    END OF GENIUS SESSION
```

#221144

Here are a few excerpts from the sample report just defined.

RUN DATE 02/24/82 PAGE NO. 1

EXAMPLE 1

PAYROLL INFORMATION BY EMPLOYEE.

LAST NAME	PAYROLL MONTH	GROSS PAY SUMMED BY EMPLOYEE
AMEER	01/31/79	$ 5,200.00
	02/28/79	$ 3,000.00
	03/30/79	$ 2,500.00
	04/30/79	$ 4,800.00
	05/31/79	$ 4,200.00
TOTAL		$ 19,700.00
BOWMAN	01/31/79	$ 3,200.00
	02/28/79	$ 3,200.00
	03/30/79	$ 5,200.00
	04/30/79	$ 3,200.00
	05/31/79	$ 3,200.00
TOTAL		$ 18,000.00
BROOKS	05/31/79	$ 1,500.00
TOTAL		$ 1,500.00
BROWN	01/31/79	$ 2,700.00
	02/28/79	$ 2,700.00
	03/30/79	$ 3,700.00
	04/30/79	$ 2,700.00
	05/31/79	$ 2,700.00
TOTAL		$ 14,500.00
CAHILL	01/31/79	$ 2,700.00
	02/28/79	$ 2,700.00
	03/30/79	$ 2,700.00
	04/30/79	$ 2,700.00
	05/31/79	$ 2,700.00
TOTAL		$ 13,500.00
CANADY	01/31/79	$ 2,000.00
	02/28/79	$ 2,000.00
	03/30/79	$ 2,000.00
	04/30/79	$ 2,000.00
	05/31/79	$ 2,000.00
TOTAL		$ 10,000.00

⋮

RUN DATE 02/24/82 PAGE NO. 10

EXAMPLE 1

PAYROLL INFORMATION BY EMPLOYEE.

LAST NAME	PAYROLL MONTH	GROSS PAY SUMMED BY EMPLOYEE
GRAND TOTAL		$458,832.30

EXIT:

1188

#221144

Chapter 7.3: The QueX Facility

The QueX facility is menu-driven. It lets you retrieve and modify data records. Instead of forming SYSTEM 2000 commands (as in the Self-Contained Facility), QueX offers a menu of records that you can retrieve or update. You can browse through the data base, selecting certain records. For each record displayed on your screen, you are given a menu of actions. Choose an action and proceed to step through the records. Actions include

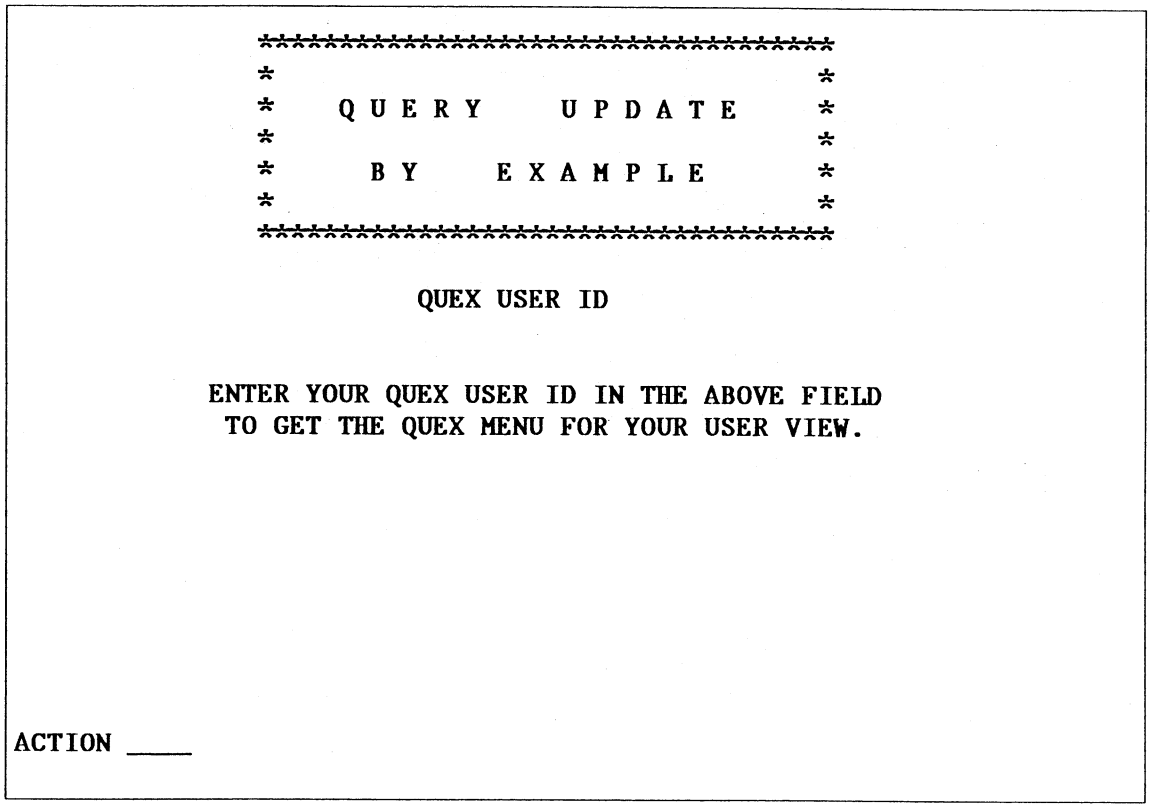
- creating new records by keying in the new values
- deleting records
- selecting records by providing the criteria in the highlighted areas on the screen
- asking for self-prompting HELP screens whenever needed.

QueX offers a fast way to browse through records of interest to you. It ensures security of information because individual views of the items in a data base are provided as appropriate. Also, since QueX does all the communicating with SYSTEM 2000 DBMS, you are free to enter volumes of data and modify the data without learning the actual SYSTEM 2000 command syntax.

QueX is most easily described by demonstrating an actual set of screens. The following pages show a portion of a QueX session. If QueX is available at your site, you can duplicate this session yourself. See your data base administrator for instructions about logging on to QueX.

A **Screen Writer** facility is also available with QueX on Sperry hardware. Screen Writer lets you implement your own screen-driven system. You design each screen and the programming logic that drives each transaction. Your code can access SYSTEM 2000 data bases, and you can edit and validate data before or after displaying a screen.

After responding to a series of logon screens, you will see the first QueX screen. Ask your data base administrator about the logon screens at your site.



Enter the name of the user view you wish to use. In our example we show an EMPLOYEE view that corresponds to the EMPLOYEE data base. However, user views can span multiple data bases.

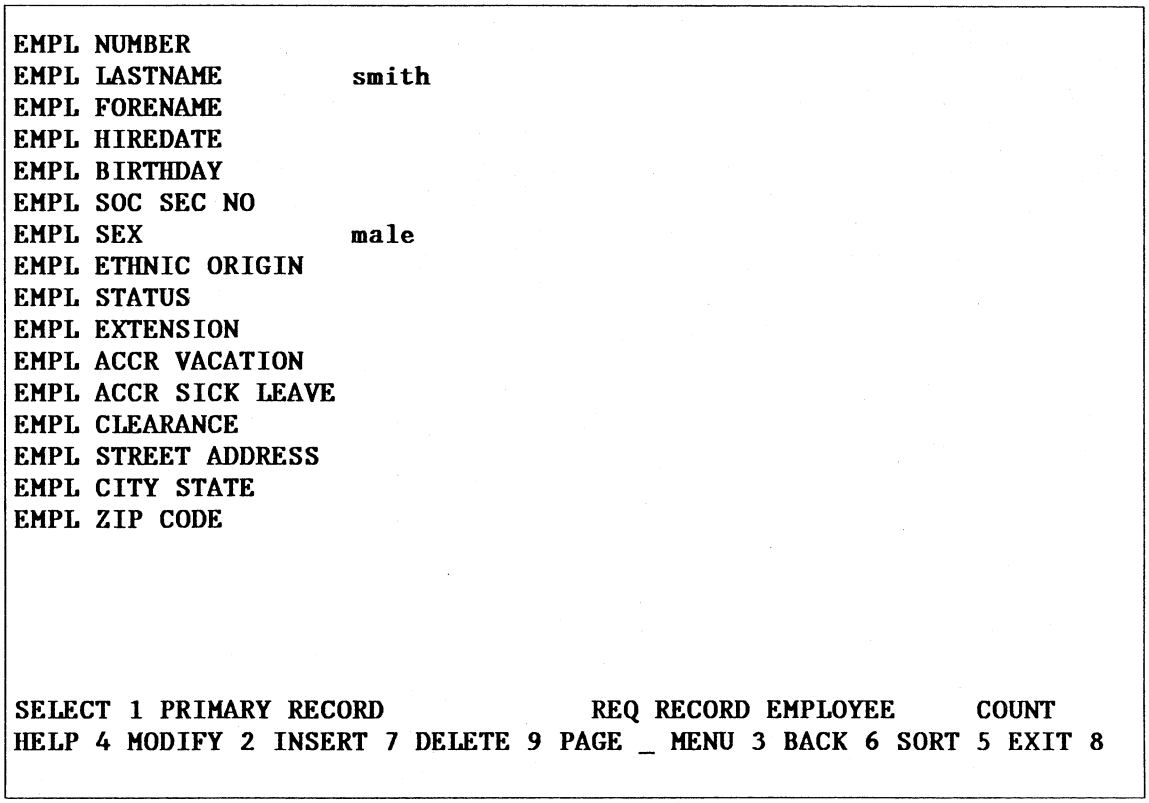
Next, a record menu screen will appear. Mark an **x** by the record you wish to see and press the ENTER key. Here, we select the EMPLOYEE record.

```
                              QUEX MENU

     FOR THE SELECTED USER VIEW THE FOLLOWING RECORDS ARE AVAILABLE.
           SELECT ONE BY PLACING AN X BEFORE THE RECORD NAME.

x EMPLOYEE          POSITION          SALARY            PAYROLL
  SKILLS            INTERESTS         EDUCATION         TRAINING




RETURN _
```

In these examples, the data you enter is shown in lowercase letters.

The EMPLOYEE menu screen appears. Notice that each item in the record is displayed in a separate row. Your action choices are across the bottom of the screen.

Suppose you want to display data about a male employee named Smith. Simply type **smith** after EMPL LASTNAME, **male** after EMPL SEX, and press Program Function key 1 (PF1) as indicated for the SELECT action at the bottom of the page.



The first selected record appears. Here, the record is for Jerry Lee Smith.

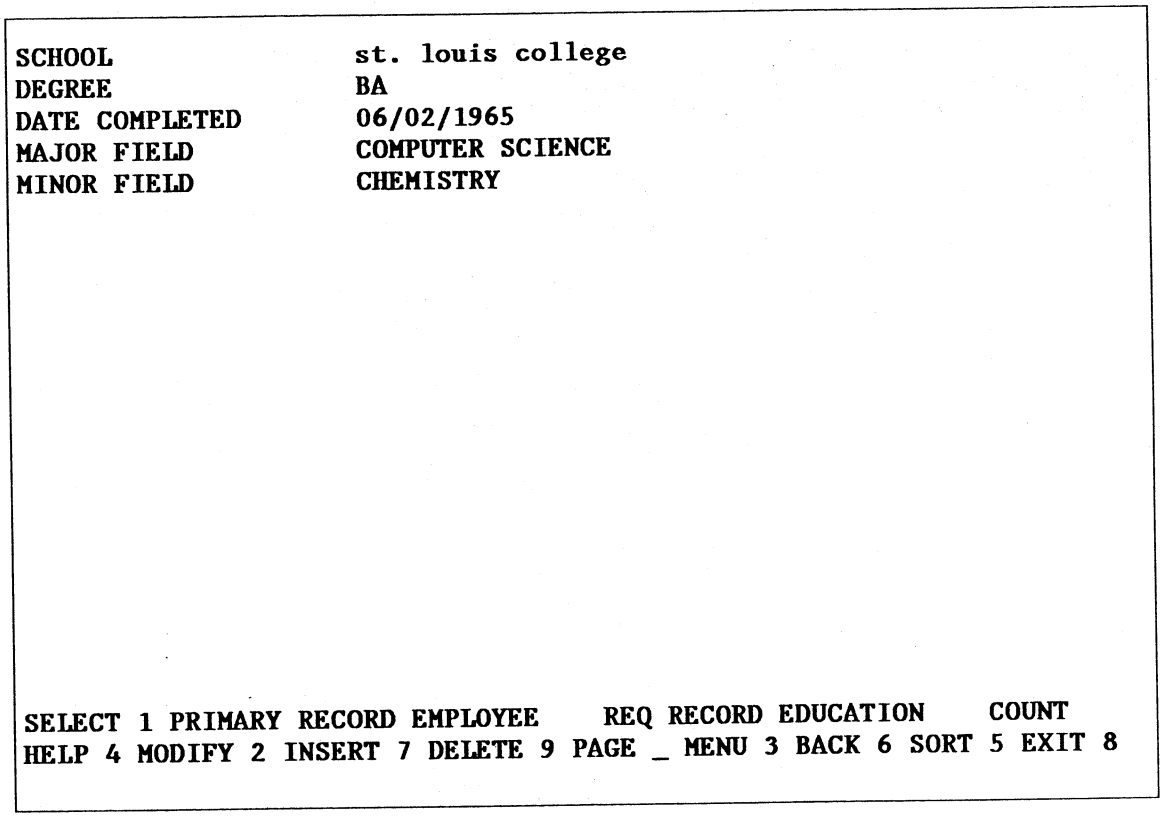
```
EMPL NUMBER           1313
EMPL LASTNAME         SMITH
EMPL FORENAME         JERRY LEE
EMPL HIREDATE         02/01/1979
EMPL BIRTHDAY         09/13/1942
EMPL SOC SEC NO       823-10-0951
EMPL SEX              MALE
EMPL ETHNIC ORIGIN    BLACK
EMPL STATUS           FULL TIME
EMPL EXTENSION        327 XT169
EMPL ACCR VACATION        .00
EMPL ACCR SICK LEAVE    16.00
EMPL CLEARANCE
EMPL STREET ADDRESS   8203 FRIAR TUCK LN.
EMPL CITY STATE       AUSTIN TX
EMPL ZIP CODE         78745




SELECT 1 PRIMARY RECORD              REQ RECORD EMPLOYEE      COUNT 0002
HELP 4 MODIFY 2 INSERT 7 DELETE 9 PAGE _ MENU 3 BACK 6 SORT 5 EXIT 8
```

The COUNT field near the bottom, right corner of the screen tells you that two records were selected. If you press the ENTER key, the next selected record appears.

In our example, suppose the first record is the one you were looking for and that you want to examine Jerry's educational history information. Simply type **education** after REQ RECORD and press the ENTER key. Jerry's first education record appears.

```
SCHOOL                ST. LOUIS UNIVERSITY
DEGREE                BA
DATE COMPLETED        06/02/1965
MAJOR FIELD           COMPUTER SCIENCE
MINOR FIELD           CHEMISTRY




SELECT 1 PRIMARY RECORD EMPLOYEE    REQ RECORD EDUCATION    COUNT
HELP 4 MODIFY 2 INSERT 7 DELETE 9 PAGE _ MENU 3 BACK 6 SORT 5 EXIT 8
```

Modifications are easy with QueX. Suppose you notice that the school name is not correct. Simply change the data on the screen and press PF2 to perform a MODIFY of the data base.



Similarly, to delete the record, press PF9. The record currently displayed and all its descendant records will be removed.

You can also use QueX for data entry. Call the menu of the record you wish to add by typing its name (say, **employee**) after REQ RECORD and pressing PF3 to display its menu.

When the menu (here, EMPLOYEE) is displayed, enter the new data after the corresponding item names and press PF7 to INSERT the new record in the data base.

```
EMPL NUMBER           1879
EMPL LASTNAME         alexander
EMPL FORENAME         jane
EMPL HIREDATE         12/14/1983
EMPL BIRTHDAY         05/13/1946
EMPL SOC SEC NO       470-50-5672
EMPL SEX              female
EMPL ETHNIC ORIGIN    caucasian
EMPL STATUS           full time
EMPL EXTENSION
EMPL ACCR VACATION
EMPL ACCR SICK LEAVE
EMPL CLEARANCE
EMPL STREET ADDRESS   12055 fire oak lane
EMPL CITY STATE       austin tx
EMPL ZIP CODE         78759




SELECT 1 PRIMARY RECORD              REQ RECORD EMPLOYEE     COUNT
HELP 4 MODIFY 2 INSERT 7 DELETE 9 PAGE _ MENU 3 BACK 6 SORT 5 EXIT 8
```

Hopefully, these examples have shown you the flavor of QueX. Of course, QueX offers other commands and much more complex functionality. See your data base administrator for more information on this facility.

Chapter 7.4: The Programming Language Extension (PLEX) Facility

The Programming Language Extension (PLEX) Facility allows users of high-level languages (COBOL, FORTRAN, and PL/I) to use SYSTEM 2000 DBMS within programs. You embed SYSTEM 2000 PLEX commands in the program to locate the records of interest in SYSTEM 2000 data bases. SYSTEM 2000 DBMS sends the data to the program area where it can be manipulated, displayed, or stored back in the data base. Multiple data bases can be accessed concurrently in one program.

While the Self-Contained Facility offers a variety of operations, you can write PLEX programs to satisfy other types of data base processing, including the following:

- accessing multiple data bases
- editing and validating input data
- loading machine-readable data directly
- looking up values in a table
- using current application programs, slightly modified by inserting PLEX commands
- using SYSTEM 2000 data with data from other sources
- using results from one query as input to subsequent queries.

The PLEX programmer can choose specific items from the data base(s) to form specific views of the data. All security features are automatically in effect with PLEX. Full selection capabilities are available through where-clauses given in the PLEX commands.

Here are some frequently used PLEX retrieval commands.

GET1 record WHERE where-clause	accesses one specified record selected by the where-clause.
LOCATE record WHERE where-clause	produces a list of selected records for use with the GET command.
GET record FIRST **GET record NEXT** **GET record LAST** **GET record PREVIOUS**	gets the first, next, last, or previous record from the list of records produced by a LOCATE command.
GETD record	gets a descendant record for the record currently in the work area.
GETA record	gets an ancestor record for the record currently in the work area.

Common PLEX commands for updating records are shown here briefly. You must get the record into your work area with a PLEX retrieval command before giving the PLEX update command.

MODIFY record	changes item values in the data base if you have modified them in your work area.
INSERT record	inserts new records into the data base.
REMOVE items	removes the item values from the record currently in the work area.
REMOVE TREE record	removes the record and its descendant records from the data base.

To set up a work area in your program, you specify a COMMBLOCK for each data base to be used. Then you specify one or more SUBSCHEMA RECORDs (SSRs) and the data base items that you are going to use in the records. COMMBLOCKs and SSRs are specified in the data declaration part of your program.

The **COMMBLOCK** has up to 15 variables, such as the data base name, password, system separator, number of records selected, subschema record name, and level of the last record selected. The COMMBLOCK also has a RETURN CODE variable that lets you check for specific error conditions.

Each declared **SUBSCHEMA RECORD** becomes a work area that holds data retrieved from the SYSTEM 2000 data base and new data to be inserted. Subschema items specified for a subschema record correspond to the items in a data base record, but you do not have to use all the items; specify only the ones needed in the PLEX program.

If you want to look at a sample PLEX program, see Appendix A.

PART 8: TIME-SAVERS AND GUIDELINES FOR AVOIDING ERRORS

This part gives timesaving techniques and guidelines for avoiding errors.

Chapter 8.1: Timesaving Techniques

Here are several timesaving techniques. They not only save you time in entering your commands, but in many cases they save actual SYSTEM 2000 processing time.

- Use **C-numbers**, for example, C102, instead of component names to save time in entering commands.
- Use the **DITTO** (or **DI**) option to repeat a retrieval-clause or an update-clause, which means everything to the left of the word WHERE.
- Use the **SAME** (or **SA**) option to use the results of the previous where-clause and, optionally, give additional where-clause conditions.
- Use the **TALLY** (or **TA**) command to count key values, especially TALLY/ALL/ if you want just a count of the distinct values and total occurrences of the item's values. TALLY is faster than PRINT COUNT. Use COUNT to count non-key items or records.
- Use **TALLY with a where-clause** to itemize and count key item values in a subset of the data base. (This option is not available for some versions of SYSTEM 2000 DBMS.)
- Use the **LIST** (or **LI**) command to display columns of values across the screen. Columnar listings often are faster to scan than the line-by-line display from the PRINT command.
- Formulate where-clause conditions that select specific records to be viewed. The shorter display will cut down on the time for scanning volumes of output. Utilize **TALLY** and **COUNT** to help form specific selection criteria. Use the **COUNT** and **LIMIT** commands to ensure that your where-clause selects the correct number of records.
- Store frequently used commands as **strings** in the definition.
- Store frequently used **functions** in the definition.

Chapter 8.2: Avoiding Errors

User errors can occur because of spelling mistakes, incorrect component numbers, effects of previous commands, or setting options incorrectly. SYSTEM 2000 DBMS issues appropriate error messages for all syntax errors. A dotted line and a cursor usually point to the area in error in the command. On the other hand, if you receive unexpected results, you've probably forgotten that an option was set prior to the command or you've specified inappropriate selection criteria. The system recognized and processed all parts of the command correctly, but has no way of knowing that the results were not as anticipated. The following sections discuss some causes of common user errors.

Section 8.2.1: Syntax Problems

Use the ECHO ON command to help identify the following syntax errors:

- forgetting the command terminator (or using the wrong one) at the end of each command, that is, the colon, semicolon, or percent sign. (The colon is used in this manual.)
- omitting required syntax or giving required syntax in the wrong order.
- omitting parentheses, usually the right parenthesis.
- forgetting the letter C before a component number.
- misspelling component numbers, names, or keywords.
- trying to sum or calculate a nonnumeric item.
- for LIST, too many items to fit across the page (screen).
- using the wrong character for the system separator. Usually this character is the asterisk, but it might have been changed for a particular data base. Enter a simple DESCRIBE command and look at the separator displayed after the component numbers.
- using the wrong Self-Contained Facility language for a command, for example, calling DEFINE and then attempting a PRINT command. Each language has its own set of commands.

Section 8.2.2: Unexpected Results

You have many ways to control the results of commands, including the LIMIT command, ordering-clause, and where-clause criteria. Also, you can use the COUNT or TALLY commands to get an idea of the scope of data available before actually retrieving or updating. Even though your command has the correct format, you might receive results that you did not anticipate. The amount of output might be more than expected or not enough. The sequence of the output might be different than what you think it should be.

too much information

- Too much information can be caused by not being selective enough in your where-clause conditions. If you are not sure which specific records or items you want to see, try a TALLY command to see how many values exist for a key item; then choose only the appropriate values. Also, the COUNT function is handy to see how many logical entries are in the data base before issuing a PRINT ENTRY command. The PRINT ENTRY command displays every logical entry in the data base unless you specify a where-clause. If the data base contains many logical entries, the volume of output will be too long (time-consuming) to analyze visually.

- If you want to look at just the selected records and not their descendant records, give the RECORD option for PRINT and LIST commands. The default is TREE, which displays all descendant records.

- The LIST command displays values for several items in columns across the page (screen), which saves space and is easy to view. The PRINT command produces a vertical display that could be lengthy since only one item value is on each line.

- Request only the specific items you want to see rather than the full record. Looking at fewer items can save time, especially if a record contains 20 or 30 items that are of no interest to you.

not enough information

- Have you set any limits prior to this command? If so, they are still in effect. Change them or cancel them with the END LIMIT command.
- Have you given the RECORD option in a previous PRINT or LIST command? If so, descendant records will not be displayed. Give a PRINT/TREE/ command to reset the option.
- Look at the where-clause to see if you gave the correct conditions. In particular, inspect the AND and OR operators to see if they connect the proper conditions. Also, for the SPANS operator, check the range of values. For the CONTAINS operator, see if the string of characters you are searching for is correct.

ignored commands

- If a set of two or three commands seems to have been ignored, check to see if the command terminator is missing or incorrect at the end of any previous command. If you're not sure, simply enter several terminators. They will make SYSTEM 2000 DBMS ignore the faulty set of commands and begin anew with the next command that you enter.
- Perhaps you are attempting commands for a language not currently in use. For example, the DEFINE and CONTROL languages do not recognize QUEST keywords, such as PRINT, DESCRIBE, TALLY, and LIST.

incorrect deletions

- When removing values and records from the data base, remember the big difference between the REMOVE and REMOVE TREE commands. The REMOVE command removes values from existing records, but does not alter the structure of the data trees. The REMOVE TREE command removes the record structures as well as the values.

PART 9: SYSTEM 2000® REFERENCE LITERATURE

Pocket Guides

Syntax Guide
Glossary

Language Specification Manuals (LSMs)

DEFINE Language
QUEST Language, including the QUEUE language
CONTROL Language and System-Wide Commands
REPORT Language
Programming Facility (PLEX)
CREATE Facility
Administrative Facilities

User's Guides

Genius
QueX

Product Support Manuals

Messages and Codes (MAC) Manuals

Installation Instructions

SYSTEM 2000 Newsletters

APPENDIX A: SAMPLE FORTRAN PLEX PROGRAM

Chapter A.1: General Structure of a PLEX Program

This chapter shows a skeleton PLEX program that outlines the general structure of setting up the work areas and program logic. FORTRAN (or COBOL or PL/I) statements can appear anywhere in a PLEX program; however, executable PLEX commands appear only after the START S2K command.

First you declare all COMMBLOCK and SSR work areas in the data definition segment of the program. Then you use SYSTEM 2000 PLEX commands to access, insert, or modify data records according to the application program requirements. All PLEX commands to be executed in the logical portion of the program must be given between the START S2K and STOP S2K commands.

Multiple data bases can be open simultaneously. Each data base requires a COMMBLOCK and appropriate SSR declarations to set up the work areas for the data base. The COMMBLOCK holds variables containing information that can be used during program execution. One of the most important variables is RETURN CODE, which should be checked after each PLEX command to see if any error occurred.

After you finish coding the source statements, the PLEX program becomes input to the PLEX processor. The PLEX processor expands each PLEX statement into one or more FORTRAN (or COBOL or PL/I) statements. This expanded source code is then ready to be compiled and executed as an ordinary program.

Chapter A.2 shows the source code and output of a sample FORTRAN PLEX program.

Skeleton of a PLEX Program

```
COMMBLOCK(s)
     .
     .
     .
SSR(s)
     .
     .
     .

$BLOCKS.
     .
     .
     .

START S2K.
     .
     .
     .
Initialize passwords(s) in
COMMBLOCK(s)
  and OPEN data base(s)
     .
     .
     .
PLEX command(s)
     .
     .
     .
CLOSE data base(s)
     .
     .
     .
PLEX command(s)
     .
     .
     .
CLOSE    remaining    data
base(s)
     .
     .
     .
STOP S2K
     .
     .
     .
```

One COMMBLOCK must be included for each data base to be accessed.

One or more subschema records can be included to refer to one or more schema records. Each SSR refers to a single schema record.

The program must have one $BLOCKS directive after the COMMBLOCK and SUBSCHEMA declarations.

First PLEX command executed by the program.

Last PLEX command executed by the program.

Chapter A.2: Sample PLEX Program (DIRINT)

This chapter contains a sample FORTRAN PLEX program called DIRINT, which uses the EMPLOYEE data base. The source listing and the results are shown.

The DIRINT program illustrates some of the PLEX retrieval commands. It produces a Personal Interest Directory for all employees in the EMPLOYEE data base. The **LOCATE ...WHERE command** produces a list of pointers to all PERSONAL INTEREST records in the data base. The **ORDER BY command** sorts the list of pointers according to the values that were given to the INTEREST (C301) item. The **GET ...NEXT command** then retrieves one record at a time into the SSR work area. Notice the **GETA command**, used to get the ancestor record (level 0) information needed for the report output.

For each personal interest, the program displays the interest and the employee's name. Personal interests are in ascending order because the LOCATE file of pointers is sorted before getting the actual records.

This sample FORTRAN PLEX program assumes that the VS FORTRAN Release 3.0 (FORTRAN 77) compiler will be used to produce the executable code.

For more details about PLEX programs, commands, and options, see the appropriate PLEX manuals. These manuals contain complete command descriptions and many examples showing actual PLEX commands combined with program logic. Source programs, expanded source listings, and output are shown in detail.

Source Code of DIRINT for VS FORTRAN Release 3.0

```
C*********************************************************************
C                          EXAMPLE PROGRAM DIRINT                     *
C ABSTRACT:                                                           *
C    THIS IS AN EXAMPLE PROGRAM THAT PRINTS A PERSONAL INTEREST       *
C    DIRECTORY FOR THE EMPLOYEES IN THE EMPLOYEE DATA BASE.  IT USES  *
C    THE COMMANDS LOCATE, ORDER BY, GET, AND GETA.                    *
C                                                                     *
C NARRATIVE:                                                          *
C    STEP                      ACTION                               *
C    1.   INITIALIZE FOR USING SYSTEM 2000 AND DATA BASE.            *
C    2.   LOCATE AND ALPHABETICALLY SORT PERSONAL INTERESTS.         *
C    3.   PRINT HEADING.                                             *
C    4.   FOR EACH INTEREST, GET AND PRINT INTEREST AND NAME.        *
C    5.   TERMINATE THE JOB.                                         *
C*********************************************************************
C
      PROGRAM DIRINT
      COMMBLOCK/ EMPLOYEE/ SSRNAM, RCODE, FREE, LASTDR, PASSWD,
     -                     RECNUM, LFPTR, LEVEL, CTIME, CDATE,
     -                     CYCLE, SEPTOR, TERMTR, DAMAGE
C
C  SSR WITH EMPLOYEE'S LAST NAME AND FORENAME
      SSR/ ENTRY OF EMPLOYEE/ LNAME(I(4))    EQ C2,
     -                        FNAME(I(4))    EQ C3
C
C  SSR WITH EMPLOYEE'S PERSONAL INTEREST
      SSR/ PRSINT OF EMPLOYEE/ INTRST(I(8))  EQ C301
C
      $BLOCKS
C
C*****
C  THE PROGRAM VARIABLES:
C
C   BLANK    - BLANK FIELD USED IN STEP 4 WHEN COMPARING INTRST TO
C              OLDINT (CHECKING FOR DUPLICATE INTEREST)
C   DEMO     - HOLDS PASSWORD FOR ASSIGNMENT TO COMMBLOCK FIELD
C   GET,GETA,LOCATE,OPEN,ORDER   - ALL HOLD NAMES OF COMMANDS; SENT AS
C              PARAMETERS TO SUBROUTINE ERRORS IF NON-ZERO RETURN CODES
C   LOOPER   - LOOP INDEX IN STEP 4
C   OLDINT   - HOLDS PREVIOUS VALUE FOR INTRST, USED IN STEP 4 TO
C              CHECK FOR DUPLICATE INTERESTS
C*****
C
```

Source Code of DIRINT for VS FORTRAN Release 3.0 (continued)

```
      EXTERNAL ERRORS
      COMMON /COMMND/ COMMND
      CHARACTER*4  DEMO, GET, GETA, LOCATE, OPEN, ORDER, COMMND
      INTEGER OLDINT(8),BLANK,LOOPER
C
      DATA BLANK /Z40404040/
      DATA DEMO /'DEMO'/
      DATA GET /'GET '/
      DATA GETA /'GETA'/
      DATA LOCATE /'LOC '/
      DATA OPEN /'OPEN'/
      DATA ORDER /'ODBY'/
C
C*************************************************************************
C                                                                        *
C                            TOP OF MODULE                               *
C                                                                        *
C*************************************************************************
C
C***
C STEP 1: INITIALIZE FOR SYSTEM 2000 AND OPEN DATA BASE
C***
      START S2K
          IF (S2KRTC .EQ. 0) GO TO 100
               WRITE (6,50) S2KRTC
   50          FORMAT ('1','S2KRTC = ', I4, 'ON START S2K')
               STOP S2K
               STOP
  100 CONTINUE
      FOR EMPLOYEE RC 1-127 GO TO ERRORS
      PASSWD = DEMO
      COMMND = OPEN
      OPEN EMPLOYEE
C
C***
C STEP 2: LOCATE AND SORT ALL PERSONAL INTERESTS IN THE COMPANY
C***
      COMMND = LOCATE
      LOCATE PRSINT WHERE INTRST EXISTS
          IF (RECNUM .GT. 0) GO TO 250
               WRITE (6,210)
  210          FORMAT ('1','NO RECORDS LOCATED')
               GO TO 999
  250 CONTINUE
      COMMND = ORDER
      ORDER BY INTRST, LNAME
```

Source Code of DIRINT for VS FORTRAN Release 3.0 (continued)

```
C
C***
C STEP 3: PRINT HEADINGS
C***
      WRITE (6,310)
  310 FORMAT ('1', T23, 'PERSONAL INTEREST DIRECTORY'///
     -        ' ', T2, 'INTEREST', T37, 'LAST NAME',
     -        T53, 'FORENAME'//)
C
C***
C STEP 4: FOR EACH INTEREST, GET AND PRINT INTEREST AND NAME
C***
      DO 350 LOOPER=1,RECNUM
          COMMND = GET
          GET PRSINT NEXT
          COMMND = GETA
          GETA ENTRY
          DO 332 I=1,8
              IF (INTRST(I) .NE. OLDINT(I)) GO TO 338
              IF (INTRST(I) .EQ. BLANK) GO TO 334
  332     CONTINUE
  334     WRITE (6,336) (LNAME(I), I=1,4), (FNAME(I), I=1,4)
  336     FORMAT (' ', T37, 4A4, T53, 4A4)
          GO TO 350
  338     WRITE (6,340) (INTRST(I),I=1,8), (LNAME(I),I=1,4),
     -                  (FNAME(I),I=1,4)
  340     FORMAT (' ', 8A4, T37, 4A4, T53, 4A4)
  342     DO 344 I=1,8
              OLDINT(I) = INTRST(I)
  344     CONTINUE
  350 CONTINUE
  999 CONTINUE
C
C***
C STEP 5: CLOSE DATA BASE AND STOP
C***
      CLOSE EMPLOYEE
      STOP S2K
      STOP
      END
```

1331

Source Code of DIRINT for VS FORTRAN Release 3.0 (continued)

```
      SUBROUTINE ERRORS
C ***** SUBROUTINE CALLED THROUGH FOR COMMAND FOR NON-ZERO RETURN
C       CODE: GIVES NAME OF COMMAND (COMMND) AND DISPLAYS
C       COMMBLOCK.
C
      INTEGER COMMND
      COMMON /COMMND/ COMMND
      COMMBLOCK/ EMPLOYEE/ SSRNAM, RCODE, FREE, LASTDR, PASSWD,
     -                     RECNUM, LFPTR, LEVEL, CTIME, CDATE,
     -                     CYCLE, SEPTOR, TERMTR, DAMAGE
      $BLOCKS
      WRITE (6,100) COMMND, SSRNAM, RCODE, LASTDR,
     -        PASSWD, RECNUM, LFPTR, LEVEL, CTIME,
     -        CDATE, CYCLE, DAMAGE
  100 FORMAT ('1', 'COMMBLOCK AFTER ', A4, ' COMMAND:'//
     -        T5, 'SSR/ITEM NAME:', T25, A32,/
     -        T5, 'RETURN CODE:', T25, I4,/
     -        T5, 'LAST RECORD FLAG:', T25, I4,/
     -        T5, 'PASSWORD:', T25, A4,/
     -        T5, 'NO. LOCATED RECORDS:', T25, I4,/
     -        T5, 'LOC. FILE POINTER:', T25, I4,/
     -        T5, 'LEVEL/ORDINAL:', T25, I4,/
     -        T5, 'CYCLE TIME:', T25, A8,/
     -        T5, 'CYCLE DATE:', T25, A8,/
     -        T5, 'CYCLE NUMBER:', T25, I4,/
     -        T5, 'DAMAGE FLAG:', T25, I4)
      CLOSE EMPLOYEE
      STOP S2K
      STOP
      END
```

1332

Output from DIRINT for VS FORTRAN Release 3.0

PERSONAL INTEREST DIRECTORY

INTEREST	LAST NAME	FORENAME
AFRICAN SAFARIS	POLANSKI	IVAN L.
ANTIQUE CAR RESTORATION	BOWMAN	HUGH E.
ANTIQUE FURNITURE RESTORATION	CAHILL	JACOB
	SALAZAR	YOLANDA
ARCHAEOLOGY	SCHMIDT	PENNY
ARTS & CRAFTS	RICHARDSON	TRAVIS Z.
BACKGAMMON	RODRIGUEZ	ROMUALDO R.
BACKPACKING	BOWMAN	HUGH E.
	KNAPP	PATRICE R.
	SMITH	JERRY LEE
BALLET	MILLSAP	JOEL B.
BASKETBALL	SMITH	JANET F.
BASKETWEAVING	LITTLEJOHN	FANNIE
BICYCLING	KAATZ	FREDDIE
BIRD WATCHING	BROOKS	RUBEN R.
BRIDGE	CANADY	FRANK A.
	REDFOX	RICHARD B.
CAMPING	HERNANDEZ	JESSE L.
	SAVAGE	WILLIAM D.
CANOEING	JOHNSON	BRADFORD
	KNAPP	PATRICE R.
	WATERHOUSE	CLIFTON P.
CERAMICS	SMITH	JERRY LEE
CHESS	GARRETT	OLAN M.
	REID	DAVID G.
CHILI COOKOFFS	JONES	RITA M.
	WATERHOUSE	CLIFTON P.
COLLECTS COMIC BOOKS	CAHILL	JACOB
COLLECTS CRYSTAL	REED	KENNETH D.
COLLECTS HIGH-FIRED POTTERY	MUELLER	PATSY
COLLECTS PEWTER	SALAZAR	YOLANDA
COLLECTS RARE BOOKS	WATERHOUSE	CLIFTON P.
COLLECTS RARE COINS	HERNANDEZ	JESSE L.
	RICHARDSON	TRAVIS Z.
COLLECTS STAMPS	CANADY	FRANK A.
CROSS COUNTRY BICYCLING	NATHANIEL	DARRYL
CROSS COUNTRY SKIING	JOHNSON	BRADFORD
	MILLSAP	JOEL B.

Output from DIRINT for VS FORTRAN Release 3.0 (continued)

DEEP SEA FISHING	QUINTERO	PEDRO
DISCO DANCING	SMITH	JANET F.
DOWNHILL SKIING	BOWMAN	HUGH E.
	HERNANDEZ	JESSE L.
ELECTRONICS	JUAREZ	ARMANDO
EXOTIC PLANTS	AMEER	DAVID
FIGURE SKATING	GIBSON	GEORGE J.
FISHING	SMITH	GARLAND P.
FOOTBALL	CHAN	TAI
	FREEMAN	LEOPOLD
GOLF	SCHMIDT	PENNY
HIKING	KNAPP	PATRICE R.
	SEATON	GARY
HOCKEY	KNAPP	PATRICE R.
	SMITH	JERRY LEE
HOME COMPUTERS	GOODSON	ALAN F.
HORSE BACK RIDING	SCHOLL	MADISON A.
HUNTING & FISHING	JUAREZ	ARMANDO
LITERATURE	MILLSAP	JOEL B.
LITHOGRAPHY	FAULKNER	CARRIE ANN
LONG DISTANCE RUNNING	GIBSON	MOLLY I.
METALSMITH	WATERHOUSE	CLIFTON P.
MILK CARTON BOAT MAKING	SLYE	LEONARD R.
MODEL SHIPBUILDING	THROCKMORTON	STEWART Q.
MOTORCYCLE TOURING	NATHANIEL	DARRYL
MOUNTAIN CLIMBING	POLANSKI	IVAN L.
MUSIC	THROCKMORTON	STEWART Q.
MYSTERY STORIES	BROWN	VIRGINA P.
OIL PAINTING	VAN HOTTEN	GWENDOLYN
OPERA	FERNANDEZ	SOPHIA
PAINTING	FAULKNER	CARRIE ANN
PHOTOGRAPHY	CANADY	FRANK A.
	GARCIA	FRANCISCO
	RODRIGUEZ	ROMUALDO R.
PINGPONG	GARRETT	OLAN M.
	SLYE	LEONARD R.
RACQUETBALL	JONES	RITA M.
RAISING ALLIGATORS	SHROPSHIRE	LELAND G.
RAISING CHICKENS	WILLIAMSON	JANICE L.
RAQUETBALL	COLLINS	LILLIAN
ROLLER SKATING CHAMPIONSHIPS	GOODSON	ALAN F.
SAILING	POLANSKI	IVAN L.
	RICHARDSON	TRAVIS Z.
	WATERHOUSE	CLIFTON P.

Output from DIRINT for VS FORTRAN Release 3.0 (continued)

SCIENCE FICTION	GARRETT	OLAN M.
SCULPTOR	SEATON	GARY
SKIN DIVING	SMITH	GARLAND P.
	VAN HOTTEN	GWENDOLYN
SOFTBALL	FREEMAN	LEOPOLD
	GARRETT	OLAN M.
	QUINTERO	PEDRO
SPORT CAR RALLIES	BROWN	VIRGINA P.
SQUASH	SCHOLL	MADISON A.
SWIMMING	GIBSON	GEORGE J.
	QUINTERO	PEDRO
	WAGGONNER	MERRILEE D.
TENNIS	LITTLEJOHN	FANNIE
	WILLIAMSON	JANICE L.
TRAVELING ABROAD	BROWN	VIRGINA P.
TROPICAL FISH	FERNANDEZ	SOPHIA
VOLLEY BALL	VAN HOTTEN	GWENDOLYN
WATER SKIING	BOWMAN	HUGH E.
	JOHNSON	BRADFORD
	KNIGHT	ALTHEA
	MUELLER	PATSY
WATERCOLOR PAINTING	SALAZAR	YOLANDA
WOODWORKING	JONES	MICHAEL Y.
WRESTLING	SAVAGE	WILLIAM D.
YOGA	KNIGHT	ALTHEA

INDEX

A
Accessing an existing data base 2-3
ADD command 4-2
Addition, in calculations 6-3
Ancestor 5-2
AND operator 3-14
Arithmetic
  calculations 6-2
  operators 6-3
ASSIGN command 4-1
ASSIGN TREE command 4-1
AT operator 3-15
Attributes 5-6
AVG system function 6-2
Avoiding errors 8-1

B
BLOCK format option 3-7

C
C-number 2-1
Calculations 6-2
Calling a Self-Contained Facility Language 7-1
CHANGE command 4-1, 4-3
CHAR item type 5-6
Child 5-2
Colon 2-1
Command File 7-3
Command format, general 2-1
Command terminator
  function 2-1
  site-specific 2-2
Commands ignored 8-4
Component 2-1, 5-5
  attributes 5-6
  definitions 5-5
  name 5-5, 5-6
  number 5-6
  types 5-5
COMPOSE command 7-4
CONTAINS operator 3-14
CONTROL language 7-1, 7-2
COUNT system function 6-2, 8-1, 8-3
CREATE facility 2-5, 5-10, 7-1, 7-2

Creating a new data base 2-4, 5-4
CO 3-7, 5-6

D
D option, in LIST command 3-11
Data base definition 5-1
DATA BASE NAME IS command 2-2
Data base, defined 5-1
Data File 7-3
Data record 5-1
Data tree 4-1, 5-2
DATE item type 5-6
DBN IS command 2-2
DECIMAL item type 5-6
DEFINE language 5-4, 7-1
Defining
  a new data base 5-1, 5-4
  functions 6-1
  items 5-5
  records 5-5, 5-8
  strings 6-1
Definition order 5-9
Descendant 5-2
DESCRIBE command 3-2
DITTO command 3-18, 8-1
Division, in calculations 6-3
DOUBLE SPACE format option 3-7

E
ECHO ON command 2-2, 8-2
END LIMIT command 3-5, 3-18
ENTRY 3-7, 5-2, 5-6
EQ operator 3-14
EXISTS operator 3-14
EXIT command 2-2

F
FAILS operator 3-14
Family 5-3
Format of commands 2-1
Format options 3-7
FORTRAN PLEX program example A-1
FTODAY system string 6-3
Function 5-1, 6-1, 6-2
Function number 6-1

G
GE operator 3-14

Genius facility 7-4
GT operator 3-14

H
HAS operator 3-15
HIGH operator 3-17
Hints 8-1

I
IN record 5-6, 5-8
Incorrect deletions 8-4
INDENT format option 3-7
Indexes 5-7
Input files, to SYSTEM 2000 DBMS 7-3
INSERT TREE command 4-1, 4-5, 5-10
Inserting new records 4-5
INTEGER item type 5-6
Item 5-1
   defined 1-1
   size 5-7
   types 3-2, 5-6

K
KEY 5-6, 5-7

L
L(w) option, in LIST command 3-11
LE operator 3-14
Level 5-3, 5-9
LIMIT command 3-5, 3-18, 8-1
LIST command 3-1, 3-10, 8-2, 8-3
LOAD command 5-10
Loading data 5-10
Log off computer system 2-2
Log on computer system 2-2
Logical entry 5-2
LOW operator 3-17
LSMs 9-1
LT operator 3-14

M
MAP command 2-4, 5-4
Master password 2-4
MAX system function 6-2
Message File 7-3
Methods for loading data 5-10
MIN system function 6-2
MONEY item type 5-6

Multiplication, in calculations 6-3

N
NAME format option 3-7
Naming a new data base 2-4
NDB IS command 2-4
NE operator 3-14
NEW DATA BASE IS command 2-4
NON-KEY 5-6, 5-7
Not enough information 8-4
NOT operator 3-15
NULL format option 3-7
NULL SUPPRESS format option 3-7
NUMBER format option 3-7

O
OB operator 3-17
OR operator 3-14
Order of definitions 5-9
ORDERED BY operator 3-17
Ordering your output 3-17
Ordering-clause 3-17
Output files, from SYSTEM 2000 DBMS 7-3
Overview of SYSTEM 2000 DBMS 1-1

P
Padding 5-6, 5-8
Parameters, in strings and functions 6-4
Parent 5-2
Passwords 2-4, 7-2
Path 5-3
Picture 5-7
PLEX facility 5-10, 7-16, A-1
PLEX program example A-1
PRINT command 3-1, 3-7, 8-3
Programming Language Extension (PLEX) facility 5-10, 7-16, A-1

Q
Query Update by Example (QueX) 7-9
QUEST language 7-1
QUEUE language 7-1, 7-2
QueX facility 5-10, 7-8

R
R(w) option, in LIST command 3-11
Record 5-1
  defined 1-1
Record definitions 5-8

RECORD format option 3-7, 8-3, 8-4
Reference literature 9-1
REMOVE command 4-1, 4-4
REMOVE TREE command 4-1, 4-6, 8-4
Removing records 4-6
REPORT command 7-4
Report File 7-3
REPORT language 7-1, 7-4
Retrieving information 3-1

S
SAME operator 3-15, 3-16, 8-1
Selecting data 3-14
Self-Contained Facility sessions 2-2
Sessions, Self-Contained Facility 2-2
Sibling 5-2
SIGMA system function 6-2
SINGLE SPACE format option 3-7
Sorting your output 3-17
SPANS operator 3-14
String 5-1, 6-1
String number 6-1
STUB format option 3-7
STUB SUPPRESS format option 3-7
Subtraction, in calculations 6-3
Subtree 5-2
SUM system function 6-2
Syntax errors 8-2
System functions 6-2
System separator 8-2
SYSTEM 2000 reference literature 9-1
System-wide commands 7-3

T
TALLY command 3-1, 3-5, 8-1, 8-3
TERMINATE command 7-2
Terminology 5-1
Time-savers 5-7, 8-1, 8-3
TITLE option, in LIST command 3-11
Too much information 8-3
Tree 5-2
TREE format option 3-7, 8-3

U
Unexpected output 8-3
Update commands 4-1
USER command 2-2
User files 7-3

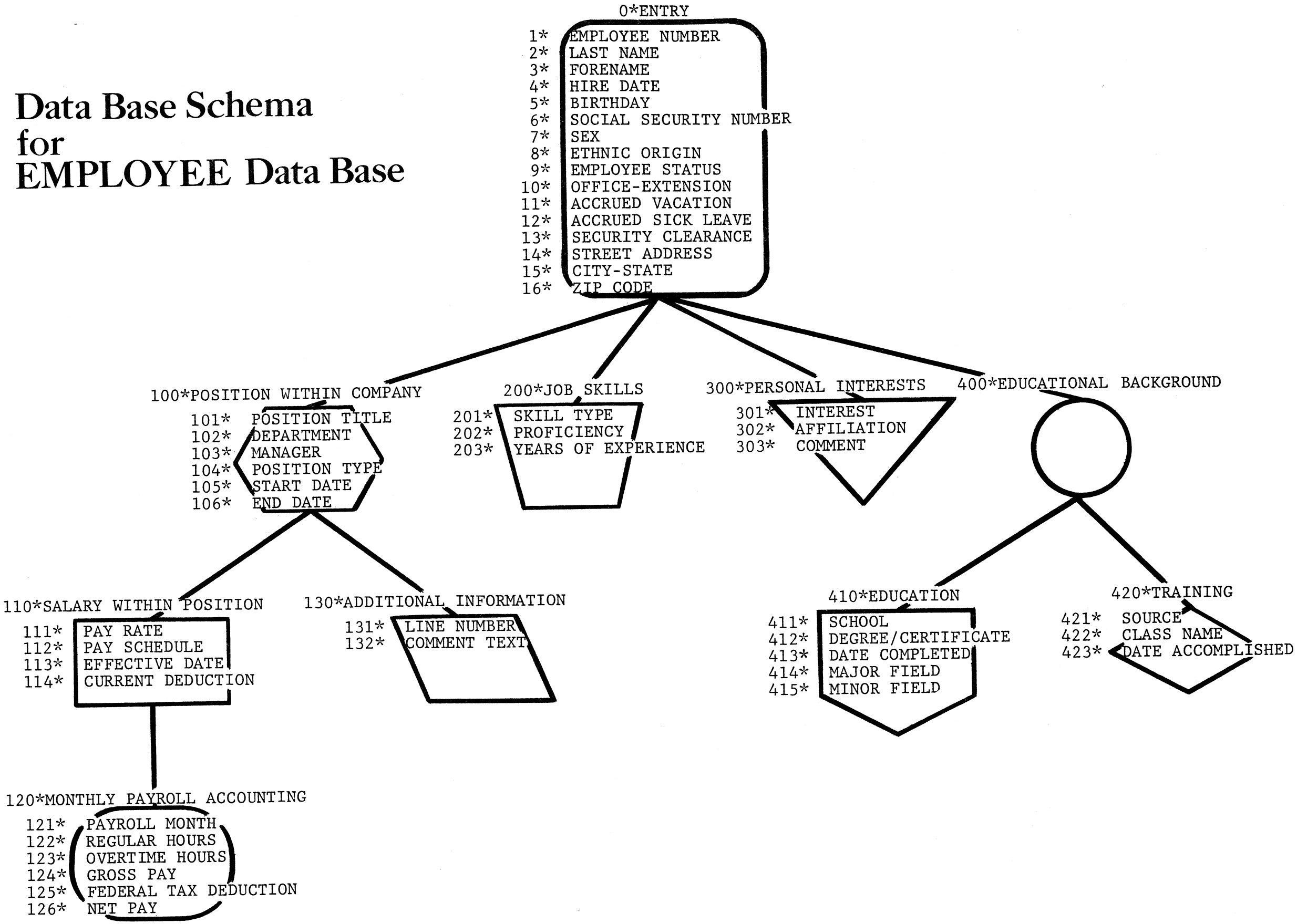
User-supplied functions 6-2

W
WHERE operator 3-14
Where-clause 3-14
WITH specification, for padding 5-8

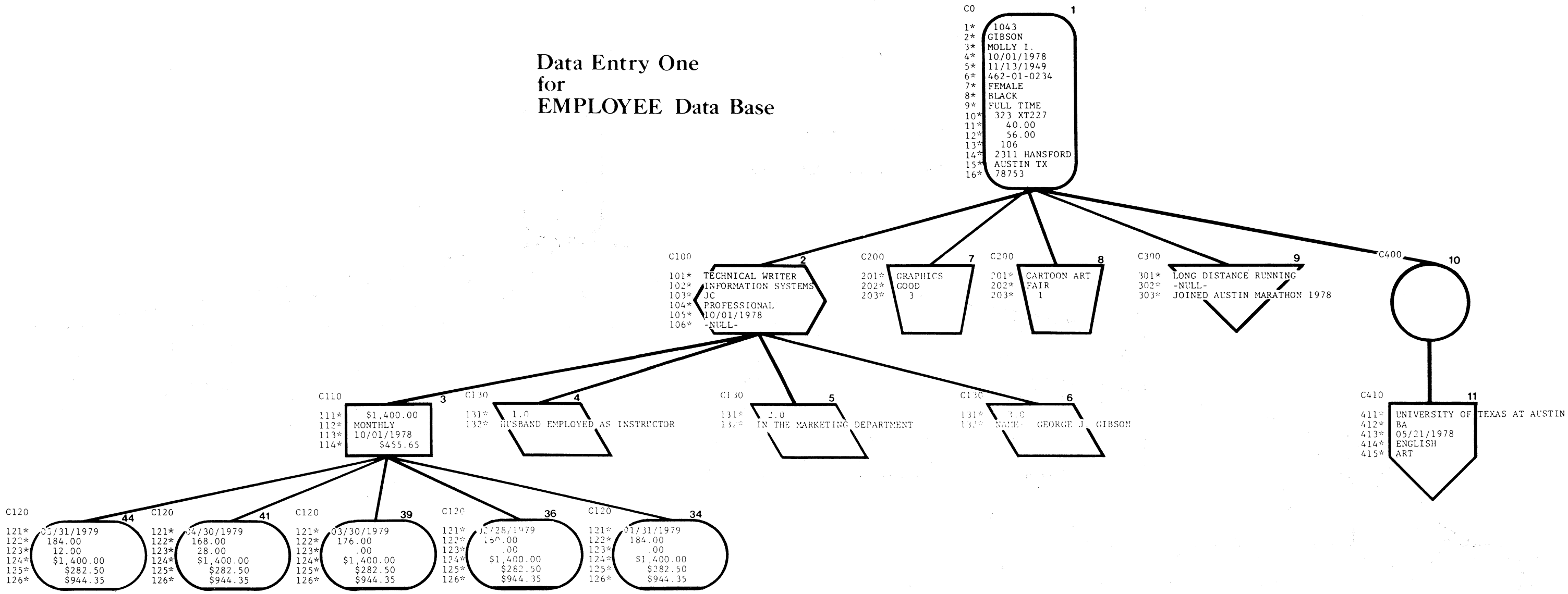
Actual DESCRIBE Output from EMPLOYEE Data Base

```
---
DESCRIBE:
SYSTEM RELEASE NUMBER  XXX
DATA BASE NAME IS      EMPLOYEE
DEFINITION NUMBER             2
DATA BASE CYCLE NUMBER       24
     1*  EMPLOYEE NUMBER (INTEGER NUMBER 9999)
     2*  LAST NAME (CHAR X(10) WITH  FEW FUTURE OCCURRENCES )
     3*  FORENAME (NON-KEY CHAR X(20))
     4*  HIRE DATE (DATE)
     5*  BIRTHDAY (DATE)
     6*  SOCIAL SECURITY NUMBER (NON-KEY CHAR X(11))
     7*  SEX (CHAR X(6) WITH MANY FUTURE OCCURRENCES )
     8*  ETHNIC ORIGIN (CHAR X(9) WITH SOME FUTURE OCCURRENCES )
     9*  EMPLOYEE STATUS (CHAR X(9) WITH MANY FUTURE OCCURRENCES )
    10*  OFFICE-EXTENSION (NON-KEY CHAR X(9))
    11*  ACCRUED VACATION (NON-KEY DECIMAL NUMBER 999.99)
    12*  ACCRUED SICK LEAVE (NON-KEY DECIMAL NUMBER 999.99)
    13*  SECURITY CLEARANCE (INTEGER NUMBER 999 WITH MANY FUTURE OCCURR
         ENCES )
    14*  STREET ADDRESS (NON-KEY CHAR X(20))
    15*  CITY-STATE (NON-KEY CHAR X(15))
    16*  ZIP CODE (CHAR X(5) WITH  FEW FUTURE OCCURRENCES )
   100*  POSITION WITHIN COMPANY (RECORD)
     101*  POSITION TITLE (NON-KEY CHAR X(10) IN 100)
     102*  DEPARTMENT (CHAR X(14) IN 100 WITH SOME FUTURE OCCURRENCES )

     103*  MANAGER (CHAR XXX IN 100 WITH  FEW FUTURE OCCURRENCES )
     104*  POSITION TYPE (CHAR X(12) IN 100 WITH SOME FUTURE OCCURRENCE
           S )
     105*  START DATE (DATE IN 100)
     106*  END DATE (NON-KEY DATE IN 100)
     110*  SALARY WITHIN POSITION (RECORD IN 100)
       111*  PAY RATE (MONEY $9999.99 IN 110)
       112*  PAY SCHEDULE (CHAR X(7) IN 110)
       113*  EFFECTIVE DATE (DATE IN 110)
       114*  CURRENT DEDUCTION (NON-KEY MONEY $9999.99 IN 110)
       120*  MONTHLY PAYROLL ACCOUNTING (RECORD IN 110)
         121*  PAYROLL MONTH (DATE IN 120)
         122*  REGULAR HOURS (NON-KEY DECIMAL NUMBER 999.99 IN 120)
         123*  OVERTIME HOURS (NON-KEY DECIMAL NUMBER 999.99 IN 120)
         124*  GROSS PAY (NON-KEY MONEY $9999.99 IN 120)
         125*  FEDERAL TAX DEDUCTION (NON-KEY MONEY $9999.99 IN 120)
         126*  NET PAY (NON-KEY MONEY $9999.99 IN 120)
     130*  ADDITIONAL INFORMATION (RECORD IN 100)
       131*  LINE NUMBER (DECIMAL NUMBER 99.9 IN 130)
       132*  COMMENT TEXT (NON-KEY TEXT X(7) IN 130)
   200*  JOB SKILLS (RECORD)
     201*  SKILL TYPE (CHAR X(12) IN 200 WITH SOME FUTURE OCCURRENCES )

     202*  PROFICIENCY (NON-KEY CHAR X(5) IN 200)
     203*  YEARS OF EXPERIENCE (NON-KEY INTEGER NUMBER 99 IN 200)
   300*  PERSONAL INTERESTS (RECORD)
     301*  INTEREST (CHAR X(12) IN 300 WITH  FEW FUTURE OCCURRENCES )
     302*  AFFILIATION (NON-KEY CHAR X(5) IN 300)
     303*  COMMENT (NON-KEY TEXT X(5) IN 300)
   400*  EDUCATIONAL BACKGROUND (RECORD)
     410*  EDUCATION (RECORD IN 400)
       411*  SCHOOL (CHAR X(15) IN 410)
       412*  DEGREE/CERTIFICATE (CHAR X(7) IN 410 WITH  FEW FUTURE OCCU
             RRENCES )
       413*  DATE COMPLETED (DATE IN 410)
       414*  MAJOR FIELD (NON-KEY CHAR X(16) IN 410)
       415*  MINOR FIELD (NON-KEY CHAR X(12) IN 410)
     420*  TRAINING (RECORD IN 400)
       421*  SOURCE (NON-KEY CHAR X(12) IN 420)
       422*  CLASS NAME (CHAR X(12) IN 420 WITH  FEW FUTURE OCCURRENCES
             )
       423*  DATE ACCOMPLISHED (DATE IN 420)
---
```

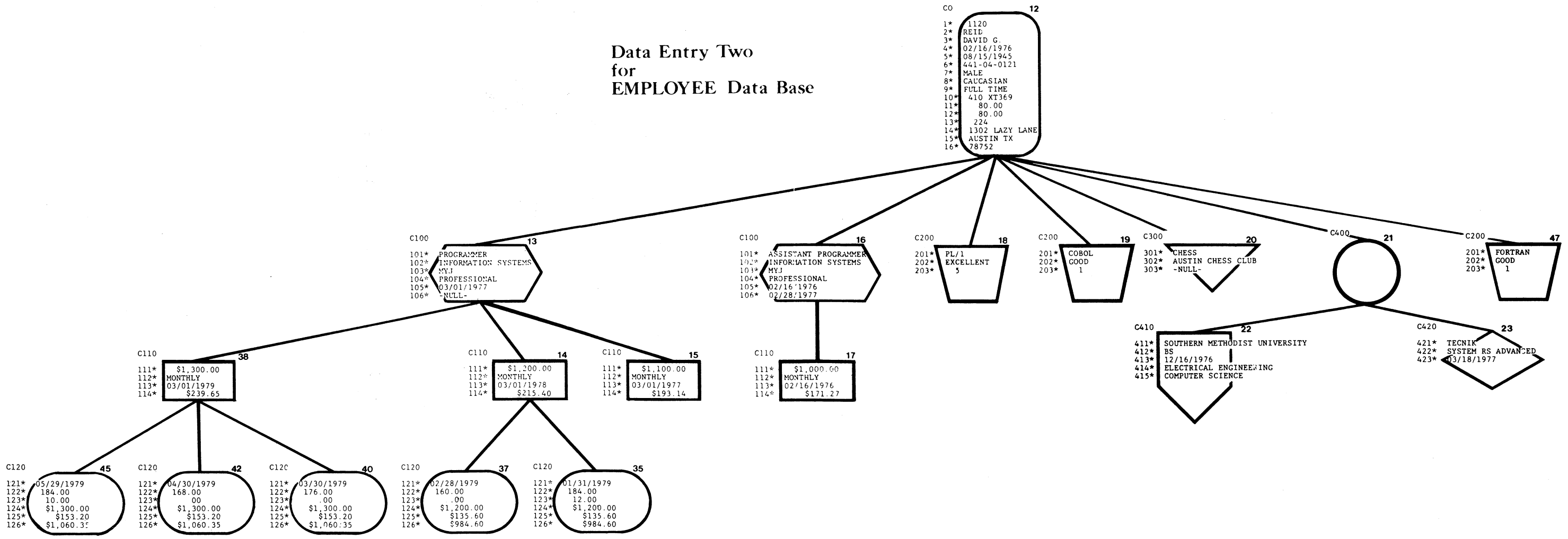
Data Base Schema for EMPLOYEE Data Base



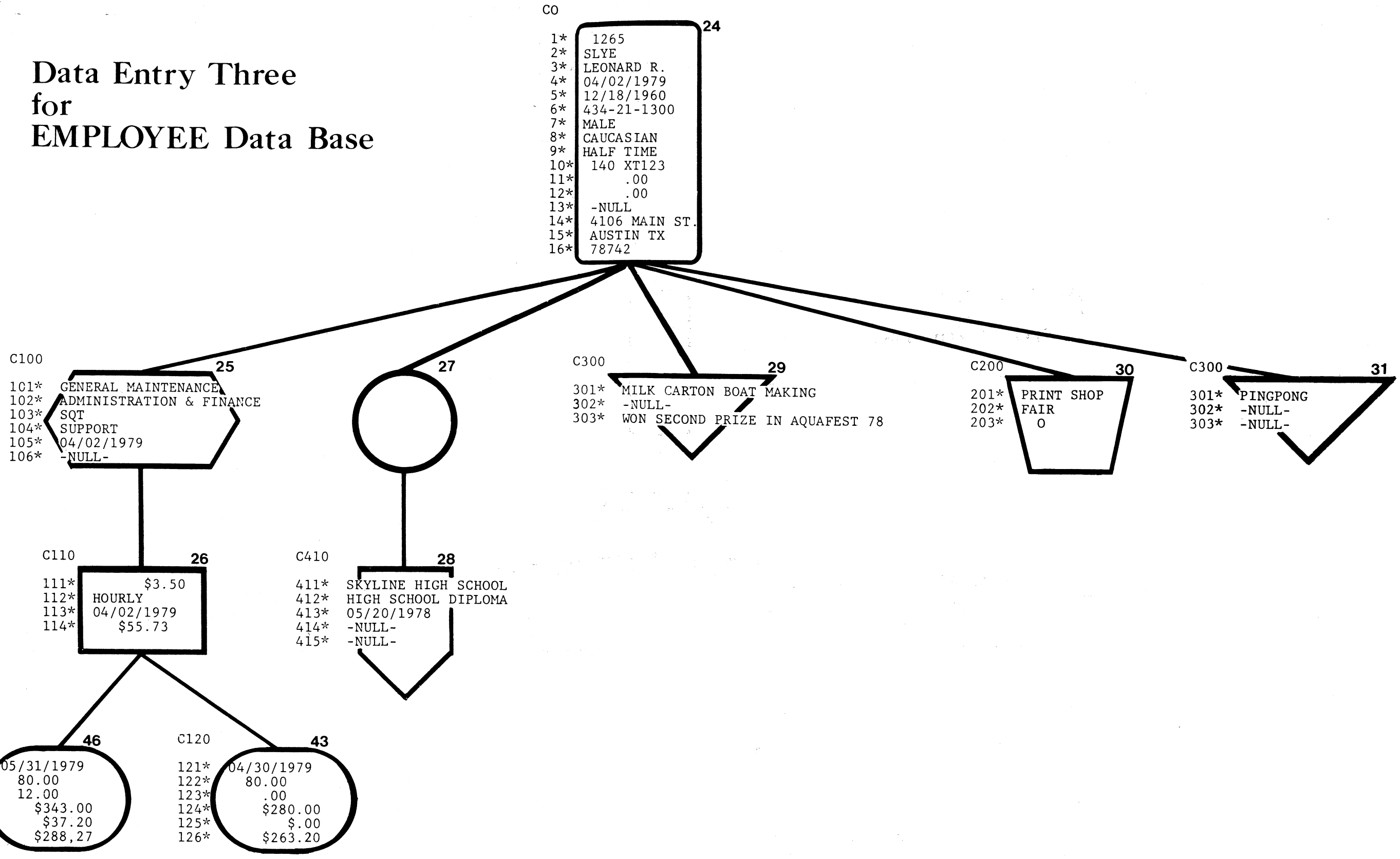
Data Entry One
for
EMPLOYEE Data Base



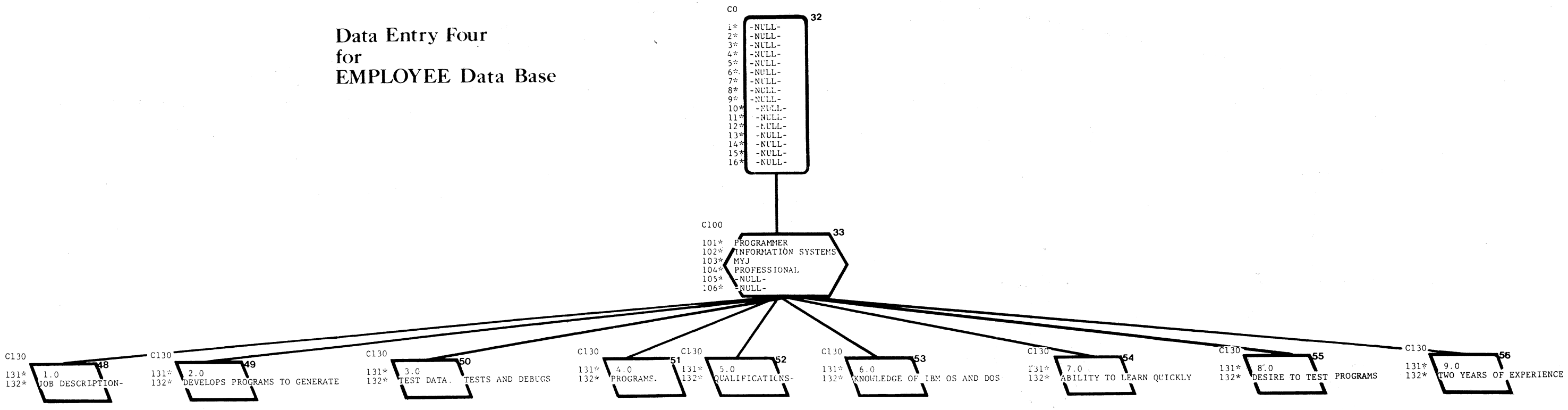
Data Entry Two
for
EMPLOYEE Data Base



Data Entry Three
for
EMPLOYEE Data Base



Data Entry Four
for
EMPLOYEE Data Base



Your Turn

If you have comments about SYSTEM 2000® or this manual, please let us know by writing your ideas in the space below. If you include your name and address, we will reply to you.

Please return this sheet to Publications Division, P.O. Box 200075, Austin, TX 78720-0075

IMPORTANT; Please include:
Manual Title: _____
Document Number: _____