SAS Publishing

SYSTEM 2000® REPORT Language

Version 12
First Edition

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 1991. *SYSTEM 2000® REPORT Language, Version 12, First Edition*. Cary, NC: SAS Institute Inc.

SYSTEM 2000® REPORT Language, Version 12, First Edition

ISBN 1-55544-181-5

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

1st printing, August 1991

Contents

Using This Book

Purpose

This document serves two purposes.

First, it serves as a guide to using the REPORT language. It describes the REPORT language in general terms, explains how you can define reports, includes complete examples with explanations, and provides some guidelines for efficient use of the REPORT language.

This book also serves as a reference manual. It includes complete descriptions of each REPORT language command.

The manual assumes that you have some experience in retrieving data from SYSTEM 2000 databases using the QUEST language (see *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition*). Although not required, some knowledge of a procedural language (such as COBOL) will help you understand the processes better.

All examples are based on the EMPLOYEE database, which is delivered with SYSTEM 2000 software. The foldout at the end of the manual includes DESCRIBE output of the EMPLOYEE database definition and diagrams of four sample logical entries.

The material in this manual pertains to running SYSTEM 2000 software on IBM mainframes (series 370 or higher, and equivalents) under MVS and CMS environments. For additional information about using SYSTEM 2000 software under CMS, see the *SYSTEM 2000 CMS Supplement, Version 12, First Edition*.

Organization of This Book

This book has three parts: Usage, Reference, and the Appendices.

Usage This part introduces the REPORT language.

Chapter 1, "Should You Use the REPORT Language?"

Chapter 2, "The Basic Process"

Chapter 3, "Defining Reports with the REPORT Language"

Chapter 4, "Executing Reports"

Reference This part provides reference information on the REPORT language.

Chapter 5, "Defining Reports"

Chapter 6, "Report-Wide Specifications"

Chapter 7, "FOR Blocks"

Chapter 8, "Specifying and Controlling Actions"

Chapter 9, "Disjoint Data Records in a Report"

Chapter 10, "The GENERATE Command"

Appendices The appendices provide information on various special areas.

Appendix A, "User Exits in the REPORT Processor"

Appendix B, "REPORT Processor Execution"

Reference Aid

This reference aid is located at the end of the book:

Index	identifies pages where specific topics and terms are discussed.

Typographical Conventions

You will notice several type styles throughout the book. Their different purposes are summarized here.

roman	is the basic type style used for most text.
UPPERCASE ROMAN	is used for references in the text to SYSTEM 2000 command keywords and database component names.
italic	is used for terms that are defined in the text, to emphasize important information, and for comments in sample code.
`MONOSPACE`	is used to show examples of commands. This book uses uppercase type for SYSTEM 2000 commands. You can enter your own commands in lowercase, uppercase, or a mixture of the two.

Syntax Conventions

SYSTEM 2000 command syntax uses the following syntax notation:

UPPERCASE ROMAN	indicates keywords. They must be spelled exactly as given in the syntax shown. Abbreviations for keywords are also shown in uppercase. If you have more than one choice, the choices are stacked vertically with a bar to the left. Select only one. A keyword without brackets is a required keyword.

italic	indicates generic terms representing words or symbols that you supply. If the term is singular, supply one instance of the term. If the term is plural, you can supply a list of terms, separated by commas. If you have more than one choice, the choices are stacked vertically with a bar to the left. Select only one. A term without brackets is a required term.
[ ]	enclose an optional portion of syntax. The brackets are not part of the command syntax. Multiple choices are stacked vertically with a bar to the left. Select only one.
| (vertical bar)	in syntax, means to select one of the vertically stacked choices. If the choices are in brackets, the choice is optional; if not, a choice is required. Then continue to the next portion of syntax, shown on the top line of the command syntax. In margins, indicates a technical change in the text for the latest version of the software.
. . . (ellipsis) . .	indicates that the previous syntax can be repeated. In examples, either vertical or horizontal ellipses indicates an omitted portion of output or a command.
ƀ	indicates a significant blank in syntax or output. Generally, spaces indicate blanks, and the ƀ is used only for emphasis.
* (asterisk)	is the default system separator throughout this book.
END	is the default entry terminator word throughout this book.

Note that symbols within syntax (such as parentheses, asterisks, or commas) are required unless enclosed in brackets or specifically noted as optional.

Example Conventions

The examples show you how to use the commands and options.

Examples are shown in uppercase. However, you can enter your own SYSTEM 2000 commands in lowercase, uppercase, or a mixture of both. Comments appear in italics within parentheses.

Page Numbering Conventions

Page numbering in the Usage part is different from that in the Reference part.

- Pages in the Usage part are numbered sequentially, that is, 1, 2, 3, and so on.
- Pages in the Reference part are numbered sequentially within the chapter number, that is, 8-1, 8-2, and so on.

Both page numbering conventions are reflected in the index. For example, the index entry for the GENERATE command lists page numbers 11 and 10-1. This means that usage information for the GENERATE command is located on page 11 of the Usage part and reference information starts in the Reference part on the first page of Chapter 10.

Additional Documentation

There are many SYSTEM 2000 publications available. To receive a free *Publications Catalog*, write to the following address:

SAS Institute Inc.
Book Sales Department
SAS Campus Drive
Cary, NC 27513

The books listed below should help you find answers to questions you may have about SYSTEM 2000 software.

- *SYSTEM 2000 Introductory Guide for SAS Software Users, Version 6, First Edition* (order #A55010) provides introductory information for SYSTEM 2000 software. It also explains how to use the SAS System with SYSTEM 2000 databases and provides examples.

- *SAS/ACCESS Interface to SYSTEM 2000 Data Management Software: Usage and Reference, Version 6, First Edition* (order #A56064) documents the ACCESS procedure, the DBLOAD procedure, and the QUEST procedure for the SAS/ACCESS interface to SYSTEM 2000 software. It explains how to create SAS/ACCESS descriptor files and how to use SAS programs with SYSTEM 2000 databases. Examples are provided.

- *SYSTEM 2000 Quick Reference Guide, Version 12, First Edition* (order #A55020) contains syntax, usage rules, and examples for all DEFINE, CONTROL, QUEST, and system-wide commands. Commands are organized alphabetically within each language, and the pages are color-coded for easy reference.

- *SYSTEM 2000 DEFINE Language, Version 12, First Edition* (order #A55012) serves as a usage and reference manual for the DEFINE language. This manual explains how to create and modify a SYSTEM 2000 database definition. It includes a detailed description and examples for each DEFINE command.

- *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* (order #A55011) provides concepts and reference material for the SYSTEM 2000 QUEST language and for system-wide commands. For example, this book documents retrieval commands, such as LIST and PRINT, the where-clause, the ordering-clause, update commands, such as INSERT TREE and CHANGE value, and so on.

- *SYSTEM 2000 CONTROL Language, Version 12, First Edition* (order #A55013) describes how to use the CONTROL language to perform administrative tasks for SYSTEM 2000 databases, such as saving and restoring databases, assigning password security, specifying input and output files, and using the Coordinating Recovery feature.

- *SYSTEM 2000 PLEX Manual, Version 12, First Edition* (order #A55017) explains how to use the SYSTEM 2000 PLEX language. This manual details the use of PLEX commands and the methods of executing PLEX programs. It provides specific

COBOL, PL/I, FORTRAN, and Assembler PLEX reference material and examples. The manual assumes a working knowledge of the programming language with which you will use SYSTEM 2000 software.

- *SYSTEM 2000 Messages and Codes, Version 12, First Edition* (order #A55015) contains all messages and codes issued by SYSTEM 2000 software. Probable causes and corrective actions accompany all error messages and warnings.

- *SYSTEM 2000 Product Support Manual, Version 12, First Edition* (order #A55016) describes how to configure SYSTEM 2000 software for single-user and Multi-User environments, how to specify SYSTEM 2000 files, and how to use user exits. Also included are descriptions of the executable modules, all execution parameters, the Accounting Log, and the Diagnostic Log.

- *SYSTEM 2000 CICS Interface, Release 11.5 under IBM OS* (order #A5511) provides supplemental details for using SYSTEM 2000 software under CICS at the macro level.

- *SYSTEM 2000 Interface to CICS (Command Level), Version 12, First Edition* (order #A55018) provides supplemental details for using SYSTEM 2000 software under CICS (command level). It includes system requirements, execution parameters, and available options.

- *SYSTEM 2000 CMS Supplement, Version 12, First Edition* (order #A55019) contains techniques, options, and commands that pertain to SYSTEM 2000 software under CMS.

- *SYSTEM 2000 DBMS Interactive Report Writing with Genius, Release 11.5 under IBM OS and CMS* (order #A5577) explains how to use the interactive Genius facility for producing reports and listings of data from SYSTEM 2000 databases. This manual contains detailed reference material and examples for all Genius prompts and user responses under MVS and CMS.

- *SYSTEM 2000 DBMS QueX User's Guide, Release 11.5A under IBM CICS, TSO, and CMS* (order #A5563) explains how to use the menu-driven QueX facility to enter data into and retrieve data from SYSTEM 2000 databases. This manual includes a step-by-step tutorial and reference information for QueX sessions.

- *SYSTEM 2000 DBMS QueX DBA Guide, Release 11.5A under IBM CICS, TSO, and CMS* (order #A5588) explains how to build user views for QueX sessions. This manual includes instructions for building, rebuilding, and deleting user views, for defining and deleting links in user views, and for maintaining the QueX system.

- *Technical Report: S2-106 Multi-User Tuning Tools for SYSTEM 2000 Software, Release 11.6 under IBM OS and CMS* (order #A55001) describes the Tuning Tools feature for SYSTEM 2000 software. The Multi-User status console commands display details about thread, scratch pad, buffer, and queue usage. The Accounting and Diagnostic Log reports display information about system performance or about specific jobs and job types running in the system. Data are extracted from these logs into a SAS data set for use in reports, and they are summarized into historical SAS data sets for more general monthly, quarterly, or annual reports.

- *Technical Report: S2-107 XBUF Caching Feature in SYSTEM 2000 Software, Release 11.6 under IBM OS* (order #A55002) documents the Extended Buffer Manager (XBUF) feature, which provides several caching techniques for both single-user and Multi-User jobs. These techniques include simple "front-end" XBUF macros, with which you can take advantage of dual logging, statistics gathering, cache modeling, and caching of database files without having to employ the more complicated CACHE macro. Later, if you need to tailor caching for special applications, you can refer to the CACHE macro documentation also included in this report.

- *Technical Report: S2-110 QUEUE Language for SYSTEM 2000 Software, Release 11.6 under IBM OS and CMS* (order #A55009) documents all aspects of the QUEUE language. Used together with the QUEST language manual, this technical report provides complete information on using the QUEUE facility in SYSTEM 2000 software.

- *SAS Technical Report S2-111, SYSTEM 2000 Internals: Database Tables, COMMON Blocks, and Debug Utilities, Version 12* (order #A55022) describes the structure of the internal tables in a SYSTEM 2000 database. The report details all table entries and the fields within the entries. Sample illustrations are included. In addition, the Update Log and Rollback Log formats are given. An Appendix lists all SYSTEM 2000 COMMON Data Areas, with an index and cross reference. This material is supplemental and is not required to use SYSTEM 2000 software.

- *SYSTEM 2000 Glossary and Subject Index, Version 12, First Edition* (order #A55021) defines specific terminology in SYSTEM 2000 software publications and includes a comprehensive index by subject to all Institute publications documenting SYSTEM 2000 software.

Changes and Enhancements

Introduction

This section lists the changes and enhancements made to the REPORT language for Version 12. This information is intended for users who have previous experience with SYSTEM 2000 software.

Mixed Case Data

Version 12 accepts and recognizes keywords and component names entered in lowercase. By default, the software translates data to all uppercase before processing, except when the data come from an alternate Command File or Data File.

The one restriction with this enhancement is that all component names are uppercased by the software when you define the database. Prior to Version 12, uppercase C3 referred to component number 3 while lowercase c3 referred to component name c3. For example, it was possible for component number 1 to have a component name of c3. With Version 12, lowercase c3 will be uppercased by the software and therefore will always be recognized as component number 3.

To disable uppercase translation, you can set the execution parameter OPT043 to YES.

New Item Types

The REPORT language now accepts DOUBLE, REAL, and UNDEFINED item types. REAL and DOUBLE values are floating point numbers. UNDEFINED values contain any of the 256 EBCDIC characters, which are treated like text except that overflow is not allowed.

Usage

Chapter 1

Should You Use the REPORT Language?

Before you start on your application, consider whether you need the REPORT language to generate your report. This chapter provides you with most of the information you need to make a decision.

The first section compares the report generation capabilities of the REPORT language, the QUEST language LIST command, and PLEX programs.

The second section includes two sample reports generated with the REPORT language, as well as the commands used. Look at these to get an idea of the REPORT language commands and what they can do.

WHAT DOES THE REPORT LANGUAGE OFFER?

The capabilities of the REPORT language lie between those of the QUEST language LIST command and those of PLEX programs.

How can you tell whether PLEX, the REPORT language, or LIST is most appropriate to your needs? Consider how the three methods compare with respect to the two basic components of any report: data and format.

In all three cases, the data can come from one or more SYSTEM 2000 databases. The differences arise when you compare how much you can manipulate the data before printing them. With LIST, you can perform calculations based on the data. With the REPORT language, you can also control whether certain data should be printed. The REPORT language also allows you to call user-written routines to manipulate data. However, you may not be able to access all the records in a logical entry. With PLEX, you can do anything permitted by the individual programming language (COBOL, FORTRAN, PL/I, or Assembler). Furthermore, with PLEX, you can access all the records in logical entry and print those values wherever you want.

The same range of possibilities exists with format. LIST gives the least flexibility; PLEX gives the most.

The three examples on the following pages may help you differentiate among the three methods. They all use the same data.

Of course, your choice of method may also depend on how much time you can spend developing the report. This in turn may depend on how much time you can spend learning PLEX or the REPORT language. To use PLEX, you embed PLEX commands in a program written in COBOL, FORTRAN, PL/I, or Assembler. To use the REPORT language, you define a report with English-like procedural commands.

The first example shows the output of a standard LIST command. Notice the repetition of Department names. Notice also that Room/Phone values appear as they are stored in the database.

```
                             Directory
                             11/28/91

*                                                          Room
  Department                 Last Name    First Name       Phone
***
* CORPORATION                BOWMAN       HUGH E.        109 XT901
* CORPORATION                GARRETT      OLAN M.        212 XT208
* CORPORATION                KNAPP        PATRICE R.     222 XT123
* INFORMATION SYSTEMS        CAHILL       JACOB          435 XT426
* INFORMATION SYSTEMS        FERNANDEZ    SOPHIA         414 XT847
* INFORMATION SYSTEMS        SMITH        JANET F.       506 XT621
* MARKETING                  AMEER        DAVID          545 XT495
* MARKETING                  BROWN        VIRGINA P.     218 XT258
* MARKETING                  CHAN         TAI            292 XT331
```

The next example shows the output of a report generated with the REPORT language. Notice that Department names are not repeated; rather, each appears once at the start of a new page. Notice the presence of a title page. Finally, look at Room/Phone values; they appear as separate pieces of data.

```
        EMPLOYEE LIST

        BY DEPARTMENT
```

```
                                                   PAGE 1
Department:  CORPORATION
-----------

     Last Name      First Name      Room   Ext
     ---------      ----------      ----   ---
     BOWMAN         HUGH E.         109    901
     GARRETT        OLAN M.         212    208
     KNAPP          PATRICE R.      222    123
```

```
                                                   PAGE 2
Department:  INFORMATION SYSTEMS
-----------

     Last Name      First Name      Room   Ext
     ---------      ----------      ----   ---
     CAHILL         JACOB           435    426
     FERNANDEZ      SOPHIA          414    847
     SMITH          JANET F.        506    621
```

```
                                                   PAGE 3
Department:  MARKETING
-----------

     Last Name      First Name      Room   Ext
     ---------      ----------      ----   ---
     AMEER          DAVID           545    495
     BROWN          VIRGINA P.      218    258
     CHAN           TAI             292    331
```

The next example shows the output of a report generated with PLEX. Notice that the last name and first name values have been combined and are followed by dots. Room/Phone values again appear as separate items. Notice also that headings are underscored with an unbroken line.

```
        EMPLOYEE LIST

        BY DEPARTMENT
```

```
                                                   PAGE 1
Department:  CORPORATION

     Name                     Room    Ext
     BOWMAN, HUGH E. . . . .  109     901
     GARRETT, OLAN M.. . . .  212     208
     KNAPP, PATRICE R. . . .  222     123
```

```
                                                   PAGE 2
Department:  INFORMATION SYSTEMS

     Name                     Room    Ext
     CAHILL, JACOB . . . . .  435     426
     FERNANDEZ, SOPHIA . . .  414     847
     SMITH, JANET F. . . . .  506     621
```

```
                                                   PAGE 3
Department:  MARKETING

     Name                     Room    Ext
     AMEER, DAVID. . . . . .  545     495
     BROWN, VIRGINIA P.. . .  218     258
     CHAN, TAI . . . . . . .  292     331
```

Now that you have compared output from PLEX, the LIST command, and the REPORT language, look at two sample reports generated with the REPORT processor.

SAMPLE REPORT LANGUAGE REPORTS

These two reports illustrate what you can do with the REPORT language.

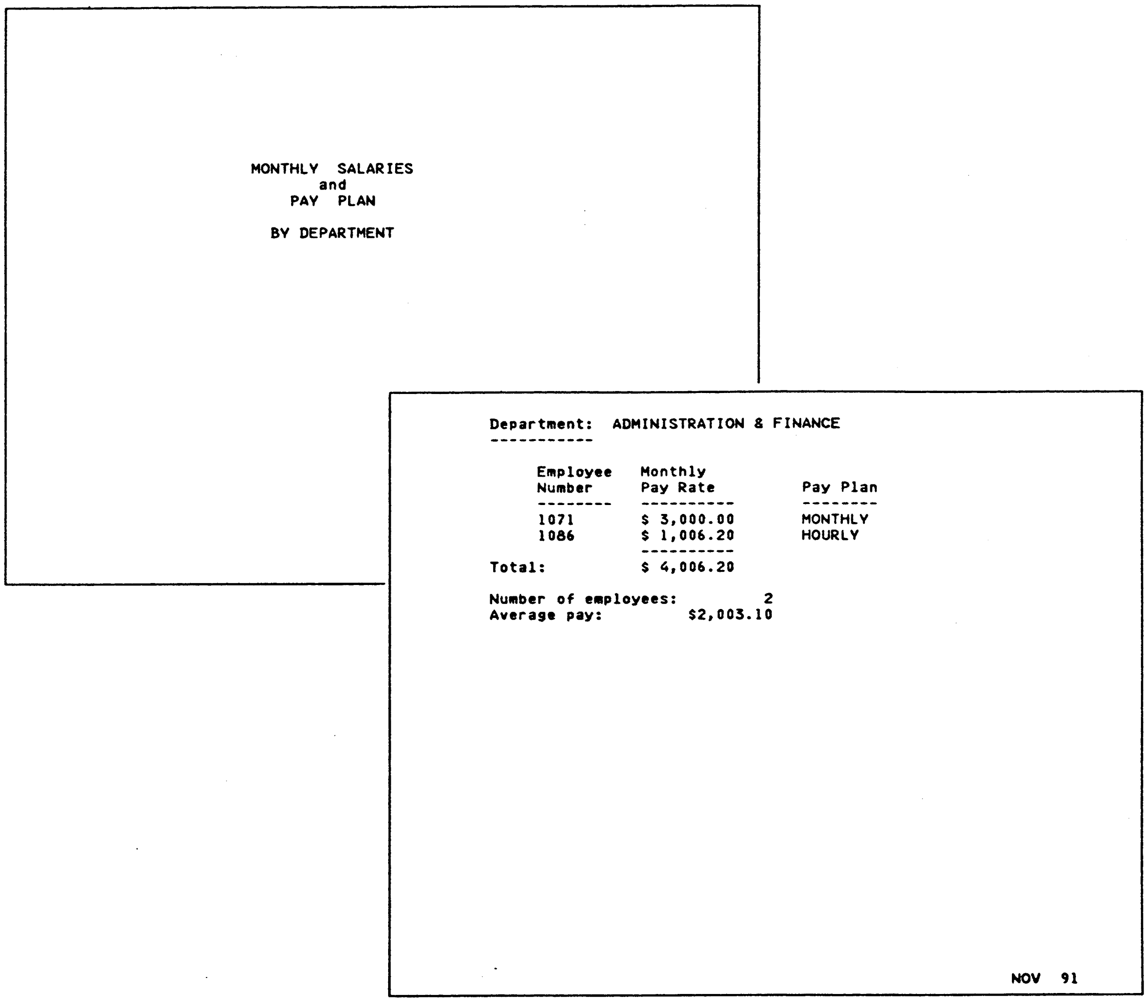
Address labels can be produced with very little effort.

```
                              VIRGINA P.   BROWN          PATSY        MUELLER          YOLANDA      SALAZAR
       ADDRESS LABELS         2713 NUTTY BROWN MILLS R*   6935 CHERRY CREEK RD.         6811 PICKET FENCE DRIVE
                              BUDA TX                78610 DRIPPING SPRINGS TX   78620  LEANDER TX             78641


GWENDOLYN    VAN HOTTEN       JESSE L.     HERNANDEZ      ALTHEA       KNIGHT           MERRILEE D.  WAGGONNER
623 FAUNTLEROY TRAIL          4319 RED STONE LANE         8222 WHITEWING WAY            941 BRIDGEWATER DRIVE
LEANDER TX             78641  ROUND ROCK TX          78664 ROUND ROCK TX         78664  AUSTIN TX              78722


OLAN M.      GARRETT          JANET F.     SMITH          HUGH E.      BOWMAN           PEDRO        QUINTERO
67 RUNNING DOE LN.            523 RIM ROCK ROAD           47 CYPRESS POINT CIRCLE       77 BUTTON QUAIL COVE
AUSTIN TX              78731  AUSTIN TX              78737 AUSTIN TX             78741  AUSTIN TX              78741


MADISON A.   SCHOLL           SOPHIA       FERNANDEZ      MICHAEL V.   JONES            PATRICE R.   KNAPP
3910 COVERED WAGON TRAIL      4700 OLD STAGE TRAIL        23 MOONLIGHT BEND LANE        19 PACK SADDLE PASS
AUSTIN TX              78741  AUSTIN TX              78744 AUSTIN TX             78748  AUSTIN TX              78748


CLIFTON P.   WATERHOUSE       MOLLY I.     GIBSON         TAI          CHAN
505 CAT MOUNTAIN TRAIL        2311 HANSFORD               1412 ARAPAHOE TRAIL             18 labels printed
AUSTIN TX              78752  AUSTIN TX              78753 AUSTIN TX             78755
```

The address labels were generated with the following REPORT statements:

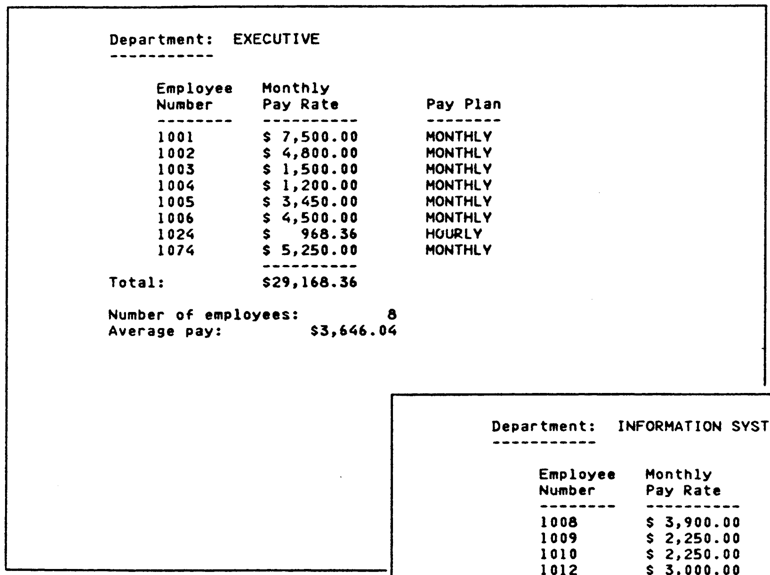
```
FOR REPORT ADRLBLS:

PHYSICAL PAGE IS 120 BY 60:
LOGICAL  PAGE IS 29 BY 5:
DECLARE PGSKIP = CNT ENTRY:

ORDER BY ZIP CODE, LAST NAME, FORENAME:

SKIP 3 LINES:
PRINT (8)'ADDRESS LABELS':

AT END,
   SKIP 3 LINES:
   PRINT R(2,ZZ9)PGSKIP, (6)'labels printed':

FOR RECORD,
     COMPUTE PGSKIP:
     IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
     SKIP 2 LINES:
     PR L(2,X(12))FORENAME, L(15,X(12))LAST NAME:
     PR L(2,X(25))STREET ADDRESS:
     PR L(2,X(20))CITY-STATE, L(24,X(5))ZIP CODE:

END REPORT:
GENERATE ADRLBLS WHERE EMPLOYEE NUMBER LT 1050:
```

This second example shows a report with various headings, calculations involving sums and averages, and summary data.

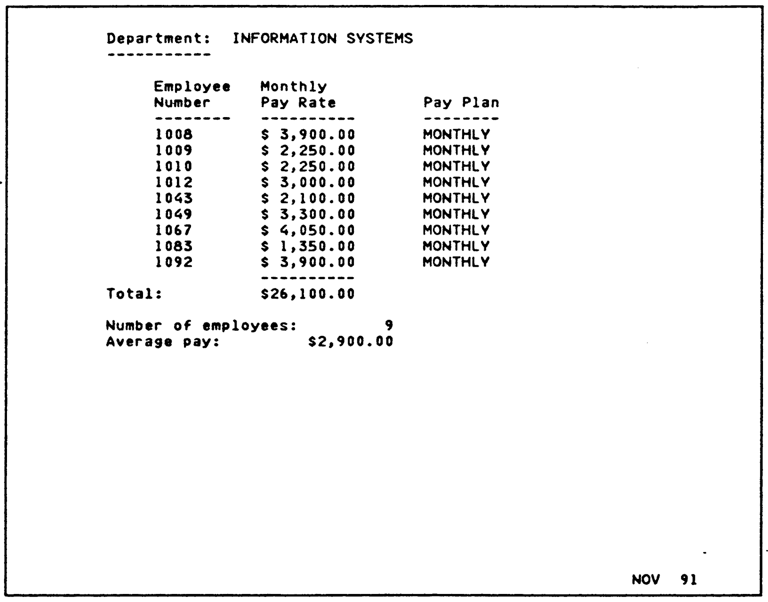


```
MONTHLY  SALARIES
       and
   PAY  PLAN

  BY DEPARTMENT
```

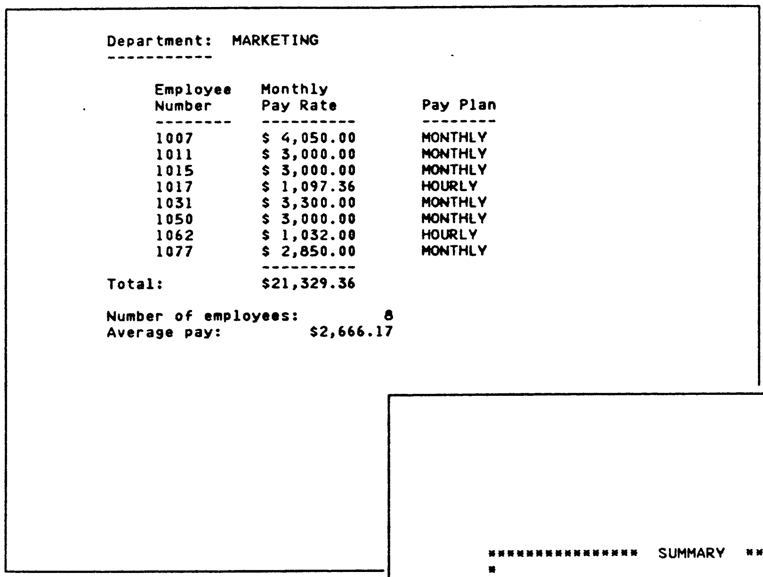
```
Department:  ADMINISTRATION & FINANCE
-----------

     Employee   Monthly
     Number     Pay Rate         Pay Plan
     --------   ----------       --------
     1071       $ 3,000.00       MONTHLY
     1086       $ 1,006.20       HOURLY
                ----------
Total:          $ 4,006.20

Number of employees:       2
Average pay:        $2,003.10




                                              NOV  91
```



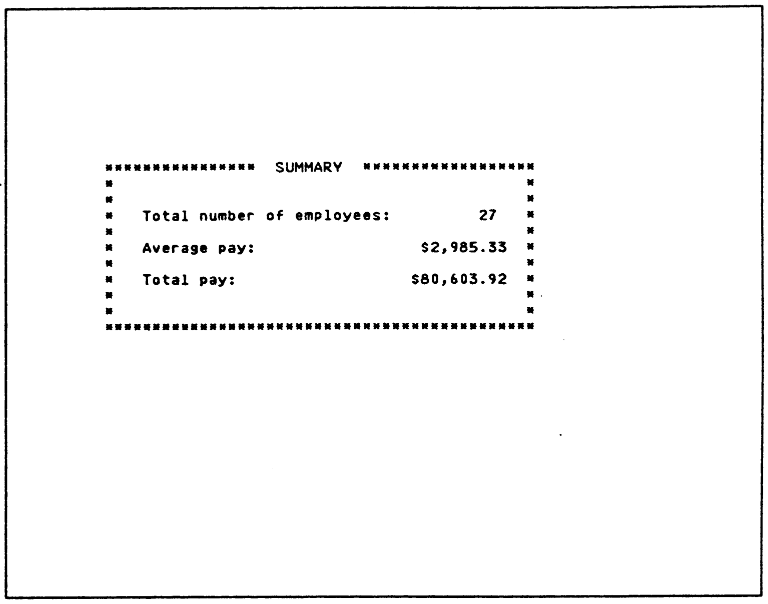
```
Department:  EXECUTIVE
-----------

     Employee   Monthly
     Number     Pay Rate         Pay Plan
     --------   ----------       --------
     1001       $ 7,500.00       MONTHLY
     1002       $ 4,800.00       MONTHLY
     1003       $ 1,500.00       MONTHLY
     1004       $ 1,200.00       MONTHLY
     1005       $ 3,450.00       MONTHLY
     1006       $ 4,500.00       MONTHLY
     1024       $   968.36       HOURLY
     1074       $ 5,250.00       MONTHLY
                ----------
Total:          $29,168.36

Number of employees:         8
Average pay:         $3,646.04
```



```
Department:  INFORMATION SYSTEMS
-----------

     Employee   Monthly
     Number     Pay Rate         Pay Plan
     --------   ----------       --------
     1008       $ 3,900.00       MONTHLY
     1009       $ 2,250.00       MONTHLY
     1010       $ 2,250.00       MONTHLY
     1012       $ 3,000.00       MONTHLY
     1043       $ 2,100.00       MONTHLY
     1049       $ 3,300.00       MONTHLY
     1067       $ 4,050.00       MONTHLY
     1083       $ 1,350.00       MONTHLY
     1092       $ 3,900.00       MONTHLY
                ----------
Total:          $26,100.00

Number of employees:         9
Average pay:         $2,900.00
```



```
Department:  MARKETING
-----------

     Employee   Monthly
     Number     Pay Rate          Pay Plan
     --------   ----------        --------
     1007       $ 4,050.00        MONTHLY
     1011       $ 3,000.00        MONTHLY
     1015       $ 3,000.00        MONTHLY
     1017       $ 1,097.36        HOURLY
     1031       $ 3,300.00        MONTHLY
     1050       $ 3,000.00        MONTHLY
     1062       $ 1,032.00        HOURLY
     1077       $ 2,850.00        MONTHLY
                ----------
Total:          $21,329.36

Number of employees:         8
Average pay:         $2,666.17
```



```
***************  SUMMARY  ****************
*                                        *
*                                        *
*   Total number of employees:       27  *
*                                        *
*   Average pay:              $2,985.33  *
*                                        *
*   Total pay:               $80,603.92  *
*                                        *
*                                        *
******************************************
```

The commands that generated this report are as follows:

```
FORMOP /ZERO/:

REPORT:
FOR REPORT PAYMON:

SUPPRESS TRUNCATION FLAG:

DECLARE PGSKIP = CNT C102:
DECLARE DATE RPDATE = (*FTODAY*):
DECLARE MONEY MNRATE = (PAY RATE * 172.00):
DECLARE MONEY TOTMND = SUM PAY RATE:
DECLARE MONEY TOTHRD = SUM MNRATE:
DECLARE MONEY TOTDPT = (TOTMND + TOTHRD):
DECLARE MONEY TOTDPS = SUM TOTDPT:
DECLARE INTEGER EMPCNT = CNT C1:
DECLARE INTEGER EMPTCN = CNT C1:
DECLARE MONEY AVGPAY = (TOTDPT/EMPCNT):
DECLARE MONEY AVGTOT = (TOTDPS/EMPTCN):
```

```
ORDER BY DEPARTMENT, EMPLOYEE NUMBER:

SKIP 10 LINES,
   PR (26)/MONTHLY  SALARIES/:
   PR (26)/       and/:
   PR (26)/    PAY  PLAN/:
   SKIP 2 LINES:
   PR (26)/  BY DEPARTMENT/:

AT END,
   SKIP 10 LINES:
   PR (11)/****************  SUMMARY  *****************/:
   PR (11)/*/,(55)/*/:
   PR (11)/*/,(55)/*/:
   PR (11)/*/,(15)/Total number of employees:/,
      R(49,ZZ9)EMPTCN,(55)/*/:
   PR (11)/*/,(55)/*/:
   PR (11)/*/,(15)/Average pay:/,R(42,$$$$,$$$.99)AVGTOT,(55)/*/:
   PR (11)/*/,(55)/*/:
   PR (11)/*/,(15)/Total pay:/,R(42,$$$$,$$$.99)TOTDPS,(55)/*/:
   PR (11)/*/,(55)/*/:
   PR (11)/*/,(55)/*/:
   PR (11)/*******************************************/:

FOR PAGE,
    SKIP 2 LINES:
    AT END,
    PR R(66,MMMBBYY)RPDATE:

FOR DEPARTMENT,
    COMPUTE PGSKIP:
    IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
    PR (11)/Department:/, L(24,X(25))DEPARTMENT:
    PR (11)/----------/:
    SKIP 2 LINES:
    PR (16)/Employee/,(27)/Monthly/:
    PR (16)/Number/,(27)/Pay Rate/,(44)/Pay Plan/:
    PR (16)/--------/,(27)/----------/,(44)/--------/:

    AT END,
       COMPUTE TOTDPS:
       PRINT (27)/----------/:
       PRINT (11)/Total:/,R(27,$ZZ,ZZZ.99)TOTDPT:
       SKIP 2 LINES:
       PRINT (11)/Number of employees:/,R(39,Z9)EMPCNT:
       PRINT (11)/Average pay:/,R(31,$$$,$$$.99)AVGPAY:
       RESET TOTHRD,TOTMND,EMPCNT:

FOR RECORD,
    IF PAY SCHEDULE EQ HOURLY* THEN
       COMPUTE TOTHRD, EMPCNT, EMPTCN,
       PR R(16,9999)EMPLOYEE NUMBER,R(27,$ZZ,ZZZ.99)MNRATE,
       L(44,XXXXXXX)PAY SCHEDULE
    ELSE
       COMPUTE TOTMND, EMPCNT, EMPTCN,
       PR R(16,9999)EMPLOYEE NUMBER,R(27,$ZZ,ZZZ.99)PAY RATE,
       L(44,XXXXXXX)PAY SCHEDULE:

END REPORT:

GENERATE PAYMON WHERE PAY RATE EXISTS AT 1 AND DEPARTMENT EXISTS AT 1
AND EMPLOYEE NUMBER LE 1100:
```

Now you have an idea of how the REPORT language compares with the LIST command and PLEX. You have also seen some different reports generated with the REPORT processor. The rest of this manual explains how you can use the REPORT language to generate your own reports.

Chapter 2

The Basic Process

Your REPORT session in general follows these steps:

1. Begin the session by issuing the REPORT command from the QUEST processor.

2. Define the reports. Each report definition specifies the database items to be included, their sort order, any calculations to be performed, and the report format. (See Chapter 5, "Defining Reports," for detailed instructions on defining reports.)

3. End your definition with the END REPORT command. Each pair of REPORT - END REPORT commands defines one report.

4. Issue the GENERATE command to execute a specific report or all reports.

   The GENERATE command can contain a where-clause to restrict the number of data records qualified for this group of reports. If there is no where-clause, a scan of the database selects the data items required for all reports to be executed. The software scans only that portion of the database needed for the report definitions.

5. When report execution is complete, the software returns you to the QUEST processor, ending the REPORT session.

6. To define another report or group of reports, reissue the REPORT command and enter the new report definitions.

Chapter 3

Defining Reports with the REPORT Language

This chapter uses examples to introduce the various features of the REPORT language.

Most of the examples shown in this manual have been run using the EMPLOYEE database, so if you can access it, you can also run them. Please note that these examples are not intended to demonstrate real situations, and, in some cases, they may not demonstrate the most efficient way of using the REPORT language. The examples were designed simply to demonstrate the software's various capabilities.

See Chapter 4, "Executing Reports," for information on how to execute the reports.

DIRECTORY REPORT

The first report originates with a LIST command. It is then enhanced by various REPORT language techniques, adding each one in sequence.

The request for the report might read something like this:

> *We need to produce a monthly report showing the employees' names, room numbers, and phone extensions grouped by department. Include the current month.*

Start with a LIST command to see what the data look like.

```
LIST /TITLE D(30)DIRECTORY,L(24)DEPARTMENT,L(10)LAST NAME,
L(11)FIRST NAME,L(9)ROOM PHONE/ DEPARTMENT, LAST NAME, FORENAME,
OFFICE-EXTENSION, ORDERED BY DEPARTMENT, LAST NAME, FORENAME
WHERE EMPLOYEE NUMBER EQ 1100 * 1225 AND DEPARTMENT EXISTS AT 1:
```

```
                             DIRECTORY
                             11/28/1991
* DEPARTMENT                 LAST NAME    FIRST NAME    ROOM PHONE
***
* ADMINISTRATION & FINANCE   JONES        RITA M.       127 XT271
* ADMINISTRATION & FINANCE   KAATZ        FREDDIE       243 XT387
* EXECUTIVE                  FAULKNER     CARRIE ANN    132 XT417
* EXECUTIVE                  NATHANIEL    DARRYL        118 XT544
* INFORMATION SYSTEMS        FREEMAN      LEOPOLD       436 XT604
* INFORMATION SYSTEMS        JOHNSON      BRADFORD      244 XT446
* INFORMATION SYSTEMS        REED         KENNETH D.    523 XT307
* INFORMATION SYSTEMS        REID         DAVID G.      410 XT369
* INFORMATION SYSTEMS        SEATON       GARY          131 XT545
* MARKETING                  GOODSON      ALAN F.       323 XT512
* MARKETING                  JUAREZ       ARMANDO       506 XT987
* MARKETING                  RICHARDSON   TRAVIS Z.     243 XT325
* MARKETING                  RODRIGUEZ    ROMUALDO R.   243 XT874
* MARKETING                  WILLIAMSON   JANICE L.     218 XT802
```

Now use the REPORT language to create the same report.

First, start the REPORT session by issuing this QUEST language command:

```
REPORT:
```

Because the REPORT processor can generate various reports at once, each report must have a name. The first version of this report becomes DIRV01.

```
FOR REPORT DIRV01,
```

Tell the REPORT processor how the data are to be ordered:

```
ORDER BY DEPARTMENT, LAST NAME, FORENAME:
```

For each record, the REPORT processor is to print the department, last name, first name, and office-extension:

```
FOR RECORD,
    PRINT DEPARTMENT, LAST NAME, FORENAME, OFFICE-EXTENSION:
```

The REPORT processor now has enough information to generate a report; end the report definition with the END REPORT command.

Your report definition appears as follows:

```
REPORT:
FOR REPORT DIRV01,
    ORDER BY DEPARTMENT, LAST NAME, FORENAME:
FOR RECORD,
    PRINT DEPARTMENT, LAST NAME, FORENAME, OFFICE-EXTENSION:
END REPORT:
```

To actually generate the report, issue the command

```
GENERATE DIRV01 WHERE EMPLOYEE NUMBER EQ 1100 * 1225
    AND DEPARTMENT EXISTS AT 1:
```

The where-clause used is the same one used in the LIST command above. From now on, the same where-clause will be used in this report.

Before running this report, look at the term *record*. The PRINT command above includes items from two schema records (DEPARTMENT belongs to C100, the other items belong to C0). However, FOR RECORD refers to report records.

To understand what report records are, consider how the REPORT processor processes data. Before the REPORT processor can format a report, it collects data from a database in the form of a table that looks like the output of a LIST command. Then the REPORT processor looks at that table line by line and formats the report. These lines are called records in the REPORT language.

For the rest of this manual, the term *record* refers to this report record, unless the record is specifically designated as a *schema record* or a *data record*.

Now look at the output generated by the last set of commands and compare it with the LIST command output.

```
ADMINISTRATIO*   JONES       RITA M.              127 XT271
ADMINISTRATIO*   KAATZ       FREDDIE              243 XT387
EXECUTIVE        FAULKNER    CARRIE ANN           132 XT417
EXECUTIVE        NATHANIEL   DARRYL               118 XT544
INFORMATION S*   FREEMAN     LEOPOLD              436 XT604
INFORMATION S*   JOHNSON     BRADFORD             244 XT446
INFORMATION S*   REED        KENNETH D.           523 XT307
INFORMATION S*   REID        DAVID G.             410 XT369
INFORMATION S*   SEATON      GARY                 131 XT545
MARKETING        GOODSON     ALAN F.              323 XT512
MARKETING        JUAREZ      ARMANDO              506 XT987
MARKETING        RICHARDSON  TRAVIS Z.            243 XT325
MARKETING        RODRIGUEZ   ROMUALDO R.          243 XT874
MARKETING        WILLIAMSON  JANICE L.            218 XT802
```

The asterisks in the first column indicate that the value is truncated. Each value goes in a field whose width is determined by the item's picture (shown below). Three spaces are provided between fields.

```
102*  DEPARTMENT       (CHAR  X(14)
  2*  LAST NAME        (CHAR  X(10)
  3*  FORENAME         (CHAR  X(20)
 10*  OFFICE-EXTENSION (CHAR  X(09)
```

All values appear left-justified within their fields because they are textual values (that is, CHARACTER or TEXT).

If you tried generating the above report, you got a blank page before and after the output. These are used to print title and back pages for your report, which you will learn how to do later.

Now start enhancing the report. First, deal with the repetition of department values. For each new department value, print the value once and then print the list of names associated with that department. The FOR DEPARTMENT phrase and a different version of the FOR RECORD phrase cause this change, as shown below.

```
REPORT:
FOR REPORT DIRV02,
    ORDER BY DEPARTMENT, LAST NAME, FORENAME:

FOR DEPARTMENT,
    PRINT DEPARTMENT:

FOR RECORD,
    PRINT LAST NAME, FORENAME, OFFICE-EXTENSION:

END REPORT:
```

To generate this second version of the report, issue

```
GENERATE DIRV02 WHERE EMPLOYEE NUMBER EQ 1100 * 1225
    AND DEPARTMENT EXISTS AT 1:
```

```
ADMINISTRATIO*
JONES         RITA M.                 127 XT271
KAATZ         FREDDIE                 243 XT387
EXECUTIVE
FAULKNER      CARRIE ANN              132 XT417
NATHANIEL     DARRYL                  118 XT544
INFORMATION S*
FREEMAN       LEOPOLD                 436 XT604
JOHNSON       BRADFORD                244 XT446
REED          KENNETH D.              523 XT307
REID          DAVID G.                410 XT369
SEATON        GARY                    131 XT545
MARKETING
GOODSON       ALAN F.                 323 XT512
JUAREZ        ARMANDO                 506 XT987
RICHARDSON    TRAVIS Z.               243 XT325
RODRIGUEZ     ROMUALDO R.             243 XT874
WILLIAMSON    JANICE L.               218 XT802
```

Department names are now printed on separate lines before the records for that department's employees. Notice that field widths were not changed.

Next, make two changes to the report: make headings out of the department names, and separate the room number and the phone extension. Also, add a six-character-wide left margin by inserting edit pictures in the two PRINT commands.

To make headings of the department names, have these start in column **6**, on a field that is **25** characters long. The names are to appear **L**eft-justified within that field. (As in the QUEST language, the length of the field for non-numeric values is indicated by Xs.)

```
PRINT L(6,X(25)) DEPARTMENT:
```

Have the last names start in the 10th column, on a 12-character field, and the forenames in column 24, also on a 12-character field.

To separate office numbers and phone extensions, print OFFICE-EXTENSION values twice. The first time they appear left-justified in column 38 on a three-character field. The second time they appear right-justified beginning in column 45 on a three-character field.

```
PRINT L(10,X(12)) LAST NAME, L(24,X(12)) FORENAME,
    L(38,XXX) OFFICE-EXTENSION, R(45,XXX) OFFICE-EXTENSION:
```

So, the third version of the report is

```
REPORT:
FOR REPORT DIRV03,
    ORDER BY DEPARTMENT, LAST NAME, FORENAME:

FOR DEPARTMENT,
    PRINT L(6,X(25)) DEPARTMENT:

FOR RECORD,
    PRINT L(10,X(12)) LAST NAME, L(24,X(12)) FORENAME,
          L(38,XXX) OFFICE-EXTENSION, R(45,XXX) OFFICE-EXTENSION:

END REPORT:
```

To generate the report:

```
GENERATE DIRV03 WHERE EMPLOYEE NUMBER EQ 1100 * 1225
     AND DEPARTMENT EXISTS AT 1:
```

```
ADMINISTRATION & FINANCE
    JONES         RITA M.        12*     *71
    KAATZ         FREDDIE        24*     *87
EXECUTIVE
    FAULKNER      CARRIE ANN     13*     *17
    NATHANIEL     DARRYL         11*     *44
INFORMATION SYSTEMS
    FREEMAN       LEOPOLD        43*     *04
    JOHNSON       BRADFORD       24*     *46
    REED          KENNETH D.     52*     *07
    REID          DAVID G.       41*     *69
    SEATON        GARY           13*     *45
MARKETING
    GOODSON       ALAN F.        32*     *12
    JUAREZ        ARMANDO        50*     *87
    RICHARDSON    TRAVIS Z.      24*     *25
    RODRIGUEZ     ROMUALDO R.    24*     *74
    WILLIAMSON    JANICE L.      21*     *02
```

The asterisks in the last two fields indicate that those values are truncated. Notice that the left-justified values have the asterisk at the end, while the right-justified values have the asterisk at the beginning.

To prevent the asterisks from appearing and have the numbers appear instead, insert the following command:

```
SUPPRESS TRUNCATION FLAG:
```

The SUPPRESS TRUNCATION FLAG command must occur before the ORDER BY command. See the fourth version of the command file, shown below.

Now, add labels to the report. First, print a label for the department name:

```
PRINT (6)/Department:/, L(19,X(25)) DEPARTMENT:
```

Although slashes delimit the strings here, other special characters can also be used. Notice that for strings, only the starting column is specified; the string is automatically left-justified.

Next, print the headings above the other values:

```
PRINT (10)/Last Name/,(24)/First Name/,(38)/Room/,(45)/Ext/:
PRINT (10)/---------/,(24)/----------/,(38)/----/,(45)/---/:
```

Since the headings should appear when the department name changes, place the PRINT commands for the headings after the FOR DEPARTMENT phrase. Each PRINT command causes a line skip, so the dashes will print under the column headings.

For the fourth version of the report, the command file is as follows:

```
REPORT:
FOR REPORT DIRV04,
SUPPRESS TRUNCATION FLAG:
ORDER BY DEPARTMENT, LAST NAME, FORENAME:

FOR DEPARTMENT,
    PRINT (6)/Department:/, L(19,X(25)) DEPARTMENT:
    PRINT (10)/Last Name/,(24)/First Name/,(38)/Room/,(45)/Ext/:
    PRINT (10)/---------/,(24)/----------/,(38)/----/,(45)/---/:

FOR RECORD,
    PRINT L(10,X(12)) LAST NAME, L(24,X(12)) FORENAME,
    L(38,XXX) OFFICE-EXTENSION, R(45,XXX) OFFICE-EXTENSION:

END REPORT:
```

To generate the report:

```
GENERATE DIRV04 WHERE EMPLOYEE NUMBER EQ 1100 * 1225
    AND DEPARTMENT EXISTS AT 1:
```

```
Department:  ADMINISTRATION & FINANCE
    Last Name     First Name    Room   Ext
    ---------     ----------    ----   ---
    JONES         RITA M.       127    271
    KAATZ         FREDDIE       243    387
Department:  EXECUTIVE
    Last Name     First Name    Room   Ext
    ---------     ----------    ----   ---
    FAULKNER      CARRIE ANN    132    417
    NATHANIEL     DARRYL        118    544
Department:  INFORMATION SYSTEMS
    Last Name     First Name    Room   Ext
    ---------     ----------    ----   ---
    FREEMAN       LEOPOLD       436    604
    JOHNSON       BRADFORD      244    446
    REED          KENNETH D.    523    307
    REID          DAVID G.      410    369
    SEATON        GARY          131    545
Department:  MARKETING
    Last Name     First Name    Room   Ext
    ---------     ----------    ----   ---
    GOODSON       ALAN F.       323    512
    JUAREZ        ARMANDO       506    987
    RICHARDSON    TRAVIS Z.     243    325
    RODRIGUEZ     ROMUALDO R.   243    874
    WILLIAMSON    JANICE L.     218    802
```

As mentioned on Page 16, the REPORT processor generates a blank page before the first line of output. You can use that page as a title page.

To add a two-line title about twelve lines from the top, add four lines before the FOR DEPARTMENT statement, as shown below in the fifth version of the report. The second SKIP command places "BY DEPARTMENT" on the second line after "EMPLOYEE LIST."

```
REPORT:
FOR REPORT DIRV05,
SUPPRESS TRUNCATION FLAG:
ORDER BY DEPARTMENT, LAST NAME, FORENAME:

SKIP 12 LINES:
    PRINT (20) /EMPLOYEE LIST/:
    SKIP 2 LINES:
    PRINT (20) /BY DEPARTMENT/:

FOR DEPARTMENT,
    PRINT (6)/Department:/, L(19,X(25)) DEPARTMENT:
    PRINT (10)/Last Name/,(24)/First Name/,(38)/Room/,(45)/Ext/:
    PRINT (10)/---------/,(24)/----------/,(38)/----/,(45)/---/:

FOR RECORD,
    PRINT L(10,X(12)) LAST NAME, L(24,X(12)) FORENAME,
    L(38,XXX) OFFICE-EXTENSION, R(45,XXX) OFFICE-EXTENSION:

END REPORT:
```

To generate the report:

```
GENERATE DIRV05 WHERE EMPLOYEE NUMBER EQ 1100 * 1225
    AND DEPARTMENT EXISTS AT 1:
```

```



                   EMPLOYEE LIST

                   BY DEPARTMENT


     Department:  ADMINISTRATION & FINANCE
         Last Name     First Name    Room   Ext
         ---------     ----------    ----   ---
         JONES         RITA M.       127    271
         KAATZ         FREDDIE       243    387
     Department:  EXECUTIVE
         Last Name     First Name    Room   Ext
         ---------     ----------    ----   ---
         FAULKNER      CARRIE ANN    132    417
         NATHANIEL     DARRYL        118    544
     Department:  INFORMATION SYSTEMS
         Last Name     First Name    Room   Ext
         ---------     ----------    ----   ---
         FREEMAN       LEOPOLD       436    604
         JOHNSON       BRADFORD      244    446
         REED          KENNETH D.    523    307
         REID          DAVID G.      410    369
         SEATON        GARY          131    545

     Department:  MARKETING
         Last Name     First Name    Room   Ext
         ---------     ----------    ----   ---
         GOODSON       ALAN F.       323    512
         JUAREZ        ARMANDO       506    987
         RICHARDSON    TRAVIS Z.     243    325
         RODRIGUEZ     ROMUALDO R.   243    874
         WILLIAMSON    JANICE L.     218    802
```

The report is practically complete. You could add a date to the title page and add more lines between departments. Instead, since the employees are already grouped by department, start each department on a new page and date-stamp each page.

First, to start each department on a new page, have the REPORT processor skip to a new page after it prints all the employees in a department. To do so, set up a counter by declaring the variable PGSKIP.

```
DECLARE PGSKIP = CNT DEPARTMENT:
```

As each department is processed, the software needs to compute the number of departments processed, and, if that number is higher than one, skip to a new page.

```
COMPUTE PGSKIP:
IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
```

The DECLARE command must be placed before the ORDER BY command. The sixth version of the report becomes

```
REPORT:
FOR REPORT DIRV06,
SUPPRESS TRUNCATION FLAG:
DECLARE PGSKIP = CNT DEPARTMENT:
ORDER BY DEPARTMENT, LAST NAME, FORENAME:

SKIP 12 LINES:
    PRINT (20) /EMPLOYEE LIST/:
    SKIP 2 LINES:
    PRINT (20) /BY DEPARTMENT/:

FOR DEPARTMENT,
    COMPUTE PGSKIP:
    IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
    PRINT (6)/Department:/, L(19,X(25)) DEPARTMENT:
    PRINT (10)/Last Name/,(24)/First Name/,(38)/Room/,(45)/Ext/:
    PRINT (10)/---------/,(24)/----------/,(38)/----/,(45)/---/:

FOR RECORD,
    PRINT L(10,X(12)) LAST NAME, L(24,X(12)) FORENAME,
    L(38,XXX) OFFICE-EXTENSION, R(45,XXX) OFFICE-EXTENSION:

END REPORT:
```

To generate the report:

```
GENERATE DIRV06 WHERE EMPLOYEE NUMBER EQ 1100 * 1225
    AND DEPARTMENT EXISTS AT 1:
```

```
EMPLOYEE LIST

BY DEPARTMENT
```

```
Department:  ADMINISTRATION & FINANCE
    Last Name      First Name     Room    Ext
    ---------      ----------     ----    ---
    JONES          RITA M.        127     271
    KAATZ          FREDDIE        243     387
```

```
Department:  EXECUTIVE
    Last Name      First Name     Room    Ext
    ---------      ----------     ----    ---
    FAULKNER       CARRIE ANN     132     417
    NATHANIEL      DARRYL         118     544
```

```
Department:  INFORMATION SYSTEMS
    Last Name      First Name     Room    Ext
    ---------      ----------     ----    ---
    FREEMAN        LEOPOLD        436     604
    JOHNSON        BRADFORD       244     446
    REED           KENNETH D.     523     307
    REID           DAVID G.       410     369
    SEATON         GARY           131     545
```

```
Department:  MARKETING
    Last Name      First Name     Room    Ext
    ---------      ----------     ----    ---
    GOODSON        ALAN F.        323     512
    JUAREZ         ARMANDO        506     987
    RICHARDSON     TRAVIS Z.      243     325
    RODRIGUEZ      ROMUALDO R.    243     874
    WILLIAMSON     JANICE L.      218     802
```

To complete the report, print today's date at the bottom of each page by declaring a DATE variable and assigning it today's date.

```
DECLARE DATE RPDATE =(*FTODAY*);
```

To print the date, use a FOR PAGE block. If you wanted the date to appear at the top of each page, you would use a PRINT command immediately after the FOR PAGE statement. In this example, to print the date at the bottom of every page, use an AT END phrase and a PRINT command.

```
FOR PAGE,
    AT END,
    PRINT R(50,MMMBBYY) RPDATE:
```

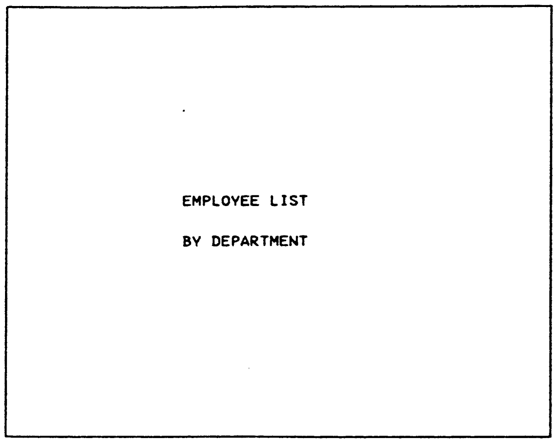
The date appears in column 50 with a three-letter month, two blanks, and a two-digit year. Because the date is numeric, it must be right-justified.

Place the FOR PAGE instructions after the report-wide instructions, that is, after the title page instructions. The seventh (and final) version of the report is as follows:

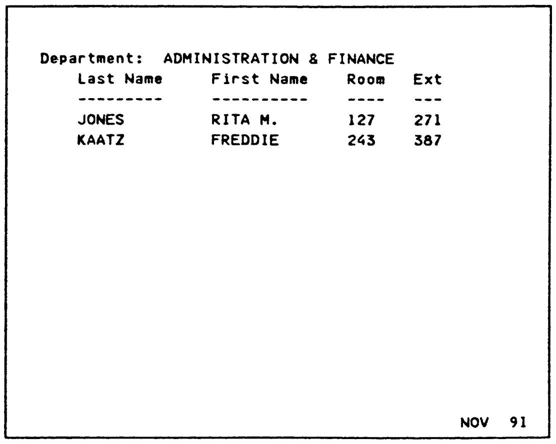
```
 REPORT:
 FOR REPORT DIRV07,
 SUPPRESS TRUNCATION FLAG:
 DECLARE PGSKIP = CNT DEPARTMENT:
DECLARE DATE RPDATE =(*FTODAY*):
ORDER BY DEPARTMENT, LAST NAME, FORENAME:

SKIP 12 LINES:
    PRINT (20) /EMPLOYEE LIST/:
    SKIP 2 LINES:
    PRINT (20) /BY DEPARTMENT/:

FOR PAGE,
    AT END,
    PRINT R(50,MMMBBYY) RPDATE:

FOR DEPARTMENT,
    COMPUTE PGSKIP:
    IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
    PRINT (6)/Department:/, L(19,X(25)) DEPARTMENT:
    PRINT (10)/Last Name/,(24)/First Name/,(38)/Room/,(45)/Ext/:
    PRINT (10)/---------/,(24)/----------/,(38)/----/,(45)/---/:

FOR RECORD,
    PRINT L(10,X(12)) LAST NAME, L(24,X(12)) FORENAME,
    L(38,XXX) OFFICE-EXTENSION, R(45,XXX) OFFICE-EXTENSION:

END REPORT:
```

To generate the report:

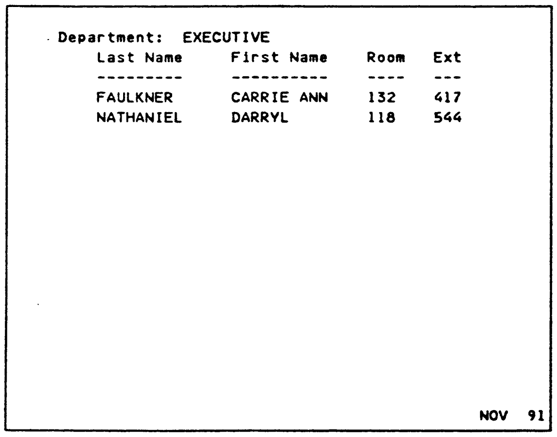
```
GENERATE DIRV07 WHERE EMPLOYEE NUMBER EQ 1100 * 1225
    AND DEPARTMENT EXISTS AT 1:
```



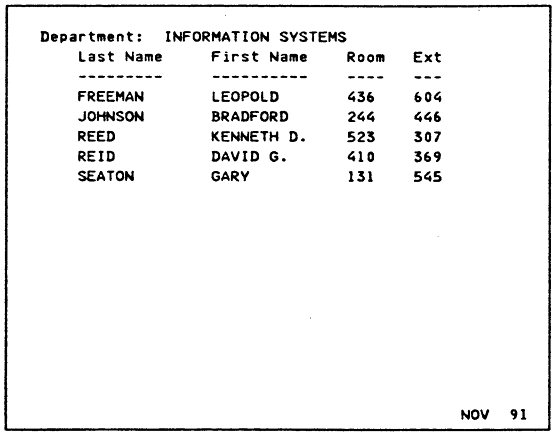
```
                 EMPLOYEE LIST

                 BY DEPARTMENT
```



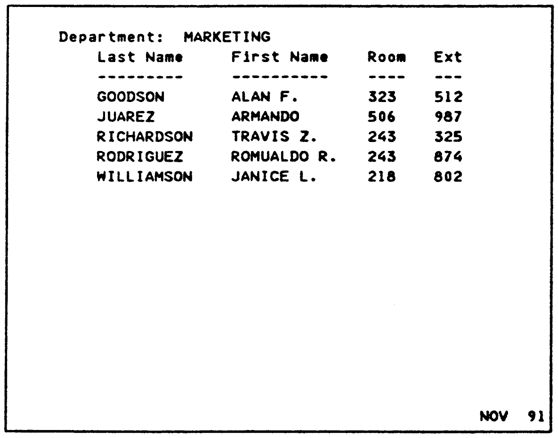
```
Department:  ADMINISTRATION & FINANCE
    Last Name     First Name     Room   Ext
    ---------     ----------     ----   ---
    JONES         RITA M.        127    271
    KAATZ         FREDDIE        243    387

                                           NOV  91
```



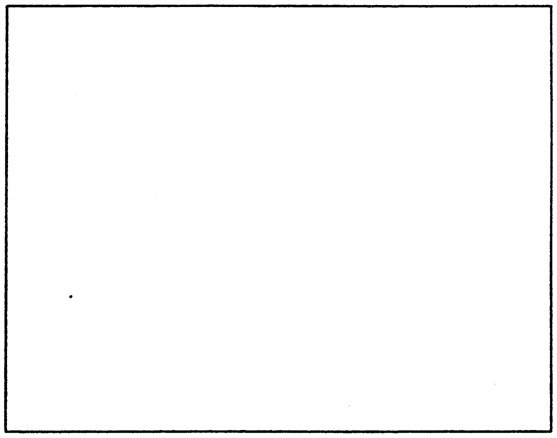
```
Department:  EXECUTIVE
    Last Name     First Name     Room   Ext
    ---------     ----------     ----   ---
    FAULKNER      CARRIE ANN     132    417
    NATHANIEL     DARRYL         118    544

                                           NOV  91
```



```
Department:  INFORMATION SYSTEMS
    Last Name     First Name     Room   Ext
    ---------     ----------     ----   ---
    FREEMAN       LEOPOLD        436    604
    JOHNSON       BRADFORD       244    446
    REED          KENNETH D.     523    307
    REID          DAVID G.       410    369
    SEATON        GARY           131    545

                                           NOV  91
```



```
Department:  MARKETING
    Last Name     First Name     Room   Ext
    ---------     ----------     ----   ---
    GOODSON       ALAN F.        323    512
    JUAREZ        ARMANDO        506    987
    RICHARDSON    TRAVIS Z.      243    325
    RODRIGUEZ     ROMUALDO R.    243    874
    WILLIAMSON    JANICE L.      218    802

                                           NOV  91
```



Notice that this report is similar to the one on Page 3.

ADDRESS LABELS REPORT

This example illustrates the use of logical pages. A logical page lets you print two or more pages of your report on one sheet of paper.

For example, you can print two 8″ by 5″ cards on one 8-1/2″ by 11″ sheet. Or you can print several 1/2″ by 3″ address labels on one sheet, as shown in this example.

For example, the request for the report might be

> *We need to produce address labels showing name, street address, city, state, and zip code. Sort the labels by zip code and name.*

As in the previous example, start by using a LIST command to see what the data look like.

```
LIST /TITLE L(10)LAST NAME, L(10)FIRST NAME, L(15)STREET,
L(15)CITY-STATE, L(5)ZIP/ FORENAME, LAST NAME, STREET ADDRESS,
CITY-STATE, ZIP CODE, ORDERED BY ZIP CODE, LAST NAME, FORENAME
WHERE EMPLOYEE NUMBER LT 1050:
```

```
* LAST NAME    FIRST NAME   STREET            CITY-STATE        ZIP
***
* VIRGINA P.   BROWN        2713 NUTTY BROW   BUDA TX           78610
                            N MILLS RD.
* PATSY        MUELLER      6935 CHERRY CRE   DRIPPING SPRING   78620
                            EK RD.            S TX
* YOLANDA      SALAZAR      6811 PICKET FEN   LEANDER TX        78641
                            CE DRIVE
* GWENDOLYN    VAN HOTTEN   623 FAUNTLEROY    LEANDER TX        78641
                            TRAIL
* JESSE L.     HERNANDEZ    4319 RED STONE    ROUND ROCK TX     78664
                            LANE
* ALTHEA       KNIGHT       8222 WHITEWING    ROUND ROCK TX     78664
                            WAY
* MERRILEE D   WAGGONNER    941 BRIDGEWATER   AUSTIN TX         78722
.                            DRIVE
* OLAN M.      GARRETT      67 RUNNING DOE    AUSTIN TX         78731
                            LN.
* JANET F.     SMITH        523 RIM ROCK RO   AUSTIN TX         78737
                            AD
* HUGH E.      BOWMAN       47 CYPRESS POIN   AUSTIN TX         78741
                            T CIRCLE
* PEDRO        QUINTERO     77 BUTTON QUAIL   AUSTIN TX         78741
                             COVE
* MADISON A.   SCHOLL       3910 COVERED WA   AUSTIN TX         78741
                            GON TRAIL
* SOPHIA       FERNANDEZ    4700 OLD STAGE    AUSTIN TX         78744
                            TRAIL
* MICHAEL Y.   JONES        23 MOONLIGHT BE   AUSTIN TX         78748
                            ND LANE
* PATRICE R.   KNAPP        19 PACK SADDLE    AUSTIN TX         78748
                            PASS
* CLIFTON P.   WATERHOUSE   505 CAT MOUNTAI   AUSTIN TX         78752
                            N TRAIL
* MOLLY I.     GIBSON       2311 HANSFORD     AUSTIN TX         78753
* TAI          CHAN         1412 ARAPAHOE T   AUSTIN TX         78755
                            RAIL
```

First, generate a report that prints the address labels in one long list. For each label, print the name on the first line, the street address on the second line, and the city, state, and zip code on the third. By using the LIST output, determine appropriate edit pictures for the values. Call this version of the report ADRLV1.

```
REPORT:
FOR REPORT ADRLV1:
ORDER BY ZIP CODE, LAST NAME, FORENAME:

FOR RECORD,
    PR L(2,X(12)) FORENAME, L(15,X(12)) LAST NAME:
    PR L(2,X(26)) STREET ADDRESS:
    PR L(2,X(20)) CITY-STATE, L(24,X(5)) ZIP CODE:

END REPORT:
```

Notice that PR is used for the PRINT command.

To generate the labels:

```
GENERATE ADRLV1 WHERE EMPLOYEE NUMBER LT 1050:
```

```
VIRGINA P.   BROWN
2713 NUTTY BROWN MILLS RD.
BUDA TX                 78610
PATSY        MUELLER
6935 CHERRY CREEK RD.
DRIPPING SPRINGS TX     78620
YOLANDA      SALAZAR
6811 PICKET FENCE DRIVE
LEANDER TX              78641
GWENDOLYN    VAN HOTTEN
623 FAUNTLEROY TRAIL
LEANDER TX              78641
     .
     .
     .
CLIFTON P.   WATERHOUSE
505 CAT MOUNTAIN TRAIL
AUSTIN TX               78752
MOLLY I.     GIBSON
2311 HANSFORD
AUSTIN TX               78753
TAI          CHAN
1412 ARAPAHOE TRAIL
AUSTIN TX               78755
```

To define logical pages, determine the physical page first. The default physical page size is 132 printing positions per line and 60 lines per page. To produce the output shown in this manual, the width was set to 120.

```
PHYSICAL PAGE IS 120 BY 60:
```

The labels must be wide enough to hold the longest line. In the example, the CITY-STATE line (with 27 characters) is the longest. A width of 29 characters per label allows for space between labels and also allows for four labels in each row.

Each label needs five lines, three for the name and address and two for space between labels.

```
    LOGICAL PAGE IS 29 BY 5:
```

These commands must be placed immediately after the FOR REPORT statement.

To have each label start on a new (logical) page, use the same method as in the previous example. (See Page 21.) Also, the name should appear on the second line of the page to allow for a blank line before each block of three lines. So skip two lines after starting the new page.

The second version of the report becomes

```
REPORT:
FOR REPORT ADRLV2:
PHYSICAL PAGE IS 120 BY 60:
LOGICAL PAGE IS 29 BY 5:
DECLARE PGSKIP = CNT ENTRY:
ORDER BY ZIP CODE, LAST NAME, FORENAME:

FOR RECORD,
    COMPUTE PGSKIP:
    IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
    SKIP 2 LINES:
    PR L(2,X(12)) FORENAME, L(15,X(12)) LAST NAME:
    PR L(2,X(26)) STREET ADDRESS:
    PR L(2,X(20)) CITY-STATE, L(24,X(5)) ZIP CODE:

END REPORT:
```

To generate the labels:

```
GENERATE ADRLV2 WHERE EMPLOYEE NUMBER LT 1050:
```

```
                              VIRGINA P.   BROWN             PATSY        MUELLER              YOLANDA      SALAZAR
                              2713 NUTTY BROWN MILLS RD.     6935 CHERRY CREEK RD.             6811 PICKET FENCE DRIVE
                              BUDA TX                 78610  DRIPPING SPRINGS TX   78620       LEANDER TX              78641


GWENDOLYN    VAN HOTTEN       JESSE L.     HERNANDEZ         ALTHEA       KNIGHT               MERRILEE D.  WAGGONNER
623 FAUNTLEROY TRAIL          4319 RED STONE LANE            8222 WHITEWING WAY                941 BRIDGEWATER DRIVE
LEANDER TX             78641  ROUND ROCK TX           78664  ROUND ROCK TX         78664       AUSTIN TX               78722


OLAN M.      GARRETT          JANET F.     SMITH             HUGH E.      BOWMAN               PEDRO        QUINTERO
67 RUNNING DOE LN.            523 RIM ROCK ROAD              47 CYPRESS POINT CIRCLE           77 BUTTON QUAIL COVE
AUSTIN TX              78731  AUSTIN TX               78737  AUSTIN TX             78741       AUSTIN TX               78741


MADISON A.   SCHOLL           SOPHIA       FERNANDEZ         MICHAEL Y.   JONES                PATRICE R.   KNAPP
3910 COVERED WAGON TRAIL      4700 OLD STAGE TRAIL           23 MOONLIGHT BEND LANE            19 PACK SADDLE PASS
AUSTIN TX              78741  AUSTIN TX               78744  AUSTIN TX             78748       AUSTIN TX               78748


CLIFTON P.   WATERHOUSE       MOLLY I.     GIBSON            TAI          CHAN
505 CAT MOUNTAIN TRAIL        2311 HANSFORD                  1412 ARAPAHOE TRAIL
AUSTIN TX              78752  AUSTIN TX               78753  AUSTIN TX             78755
```

Notice the blank label at the beginning of the list. This is the "title page" for the report. In this example it is left blank.

There is also a "last page." You can print the total number of labels in this space by using the PGSKIP variable, which counts the number of entries.

Use an AT END phrase after the ORDER BY command to specify what is to happen at the end of the report.

```
    AT END,
       SKIP 3 LINES:
       PR R(2,ZZ9) PGSKIP, (6) 'labels printed':
```

The Z is an editing character that replaces leading zeroes with blanks. The third version of the report becomes

```
   REPORT:
   FOR REPORT ADRLV3:
   PHYSICAL PAGE IS 120 BY 60:
   LOGICAL PAGE IS 29 BY 5:
   DECLARE PGSKIP = CNT ENTRY:
   ORDER BY ZIP CODE, LAST NAME, FORENAME:

    AT END,
       SKIP 3 LINES:
       PR R(2,ZZ9) PGSKIP, (6)'labels printed':

   FOR RECORD,
       COMPUTE PGSKIP:
       IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
       SKIP 2 LINES:
       PR L(2,X(12)) FORENAME, L(15,X(12)) LAST NAME:
       PR L(2,X(26)) STREET ADDRESS:
       PR L(2,X(20)) CITY-STATE, L(24,X(5)) ZIP CODE:

   END REPORT:
```

To generate these labels:

```
GENERATE ADRLV3 WHERE EMPLOYEE NUMBER LT 1050:
```

```
                              VIRGINA P.   BROWN           PATSY        MUELLER           YOLANDA      SALAZAR
                              2713 NUTTY BROWN MILLS RD.    6935 CHERRY CREEK RD.         6811 PICKET FENCE DRIVE
                              BUDA TX               78610   DRIPPING SPRINGS TX   78620   LEANDER TX            78641

GWENDOLYN    VAN HOTTEN       JESSE L.     HERNANDEZ       ALTHEA       KNIGHT            MERRILEE D.  WAGGONNER
623 FAUNTLEROY TRAIL          4319 RED STONE LANE           8222 WHITEWING WAY            941 BRIDGEWATER DRIVE
LEANDER TX            78641   ROUND ROCK TX         78664   ROUND ROCK TX         78664   AUSTIN TX             78722

OLAN M.      GARRETT          JANET F.     SMITH           HUGH E.      BOWMAN            PEDRO        QUINTERO
67 RUNNING DOE LN.            523 RIM ROCK ROAD             47 CYPRESS POINT CIRCLE       77 BUTTON QUAIL COVE
AUSTIN TX             78731   AUSTIN TX             78737   AUSTIN TX             78741   AUSTIN TX             78741

MADISON A.   SCHOLL           SOPHIA       FERNANDEZ       MICHAEL Y.   JONES             PATRICE R.   KNAPP
3910 COVERED WAGON TRAIL      4700 OLD STAGE TRAIL          23 MOONLIGHT BEND LANE        19 PACK SADDLE PASS
AUSTIN TX             78741   AUSTIN TX             78744   AUSTIN TX             78748   AUSTIN TX             78748

CLIFTON P.   WATERHOUSE       MOLLY I.     GIBSON          TAI          CHAN
505 CAT MOUNTAIN TRAIL        2311 HANSFORD                 1412 ARAPAHOE TRAIL            18 labels printed
AUSTIN TX             78752   AUSTIN TX             78753   AUSTIN TX             78755
```

Sometimes, when you are designing (or debugging) a report, you need to determine the number of a line or of a page being printed. To do so, print values of the LINE and PAGE report variables. For example, consider these changes to the report:

```
REPORT:
FOR REPORT DEBUG:
PHYSICAL PAGE IS 120 BY 60:
LOGICAL PAGE IS 29 BY 5:
DECLARE PGSKIP = CNT ENTRY:
ORDER BY ZIP CODE, LAST NAME, FORENAME:

 AT END,
    SKIP 3 LINES:
    PR R(2,ZZ9) PGSKIP, (6)'labels printed':

FOR RECORD,
    COMPUTE PGSKIP:
    IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
    PR (2) /Page=/, R(8,9) PAGE,(11) /Line=/,R(17,9)LINE:
    PR L(2,X(12)) FORENAME, L(15,X(12)) LAST NAME:
    PR L(2,X(26)) STREET ADDRESS:
    PR L(2,X(20)) CITY-STATE, L(24,X(5)) ZIP CODE:
    PR (2) /Page=/, R(8,9) PAGE,(11) /Line=/,R(17,9)LINE:

END REPORT:
```

To generate the labels:

```
GENERATE DEBUG WHERE EMPLOYEE NUMBER LT 1010:
```

```
                              Page= 1  Line= 1              Page= 2  Line= 1              Page= 3  Line= 1
                              VIRGINA P.   BROWN            YOLANDA      SALAZAR          JESSE L.     HERNANDEZ
                              2713 NUTTY BROWN MILLS RD.    6811 PICKET FENCE DRIVE       4319 RED STONE LANE
                              BUDA TX               78610   LEANDER TX            78641   ROUND ROCK TX         78664
                              Page= 1  Line= 5              Page= 2  Line= 5              Page= 3  Line= 5
Page= 4  Line= 1              Page= 5  Line= 1              Page= 6  Line= 1              Page= 7  Line= 1
ALTHEA       KNIGHT           OLAN M.      GARRETT          HUGH E.      BOWMAN           MICHAEL Y.   JONES
8222 WHITEWING WAY            67 RUNNING DOE LN.            47 CYPRESS POINT CIRCLE       23 MOONLIGHT BEND LANE
ROUND ROCK TX         78664   AUSTIN TX             78731   AUSTIN TX             78741   AUSTIN TX             78748
Page= 4  Line= 5              Page= 5  Line= 5              Page= 6  Line= 5              Page= 7  Line= 5
Page= 8  Line= 1              Page= 9  Line= 1
PATRICE R.   KNAPP            CLIFTON P.   WATERHOUSE
19 PACK SADDLE PASS           505 CAT MOUNTAIN TRAIL          9 labels printed
AUSTIN TX             78748   AUSTIN TX             78752
Page= 8  Line= 5              Page= 9  Line= 5
```

MONTHLY SALARIES REPORT

This example illustrates using the REPORT processor to perform calculations and obtain summary data.

From now on, some features of the REPORT language that have been covered before are not explained here. Also, some steps are explained by comments within the sample code.

For example, the request for the report might be

> *We need a report that shows the monthly pay and pay plan of each employee. The report should group employees by department and provide summary data for each department. At the end of the report, summarize the data for the entire company.*

As before, start by using a LIST command to see what the data look like.

```
LIST /TITLE L(25)DEPARTMENT,EMP-ID, PAY RATE, L(15)PAY PLAN/
DEPARTMENT, EMPLOYEE NUMBER, PAY RATE, PAY SCHEDULE,
ORDERED BY DEPARTMENT, EMPLOYEE NUMBER
WHERE PAY RATE EXISTS AT 1 AND DEPARTMENT EXISTS AT 1
AND EMPLOYEE NUMBER LE 1100:
```

```
* DEPARTMENT                  EMP-ID   PAY RATE   PAY PLAN
***
* ADMINISTRATION & FINANCE     1071   $3,000.00   MONTHLY
* ADMINISTRATION & FINANCE     1086       $5.85   HOURLY
* EXECUTIVE                    1001   $7,500.00   MONTHLY/BONUS
* EXECUTIVE                    1002   $4,800.00   MONTHLY/BONUS
* EXECUTIVE                    1003   $1,500.00   MONTHLY
* EXECUTIVE                    1004   $1,200.00   MONTHLY
* EXECUTIVE                    1005   $3,450.00   MONTHLY
* EXECUTIVE                    1006   $4,500.00   MONTHLY/BONUS
* EXECUTIVE                    1024       $5.63   HOURLY
* EXECUTIVE                    1074   $5,250.00   MONTHLY
* INFORMATION SYSTEMS          1008   $3,900.00   MONTHLY
* INFORMATION SYSTEMS          1009   $2,250.00   MONTHLY
* INFORMATION SYSTEMS          1010   $2,250.00   MONTHLY
* INFORMATION SYSTEMS          1012   $3,000.00   MONTHLY
* INFORMATION SYSTEMS          1043   $2,100.00   MONTHLY
* INFORMATION SYSTEMS          1049   $3,300.00   MONTHLY
* INFORMATION SYSTEMS          1067   $4,050.00   MONTHLY
* INFORMATION SYSTEMS          1083   $1,350.00   MONTHLY
* INFORMATION SYSTEMS          1092   $3,900.00   MONTHLY
* MARKETING                    1007   $4,050.00   MONTHLY/BONUS
* MARKETING                    1011   $3,000.00   MONTHLY
* MARKETING                    1015   $3,000.00   MONTHLY/COMM
* MARKETING                    1017       $6.38   HOURLY
* MARKETING                    1031   $3,300.00   MONTHLY/COMM
* MARKETING                    1050   $3,000.00   MONTHLY/COMM
* MARKETING                    1062       $6.00   HOURLY
* MARKETING                    1077   $2,850.00   MONTHLY
```

In this example, all monthly employees need to appear as MONTHLY. Therefore, to prevent '/BONUS' and '/COMM' from appearing, use a field of seven and suppress truncation.

The first version of the report (SALARV1), resembles the report used to generate the directory report.

```
REPORT:
FOR REPORT SALARV1:

SUPPRESS TRUNCATION FLAG:

DECLARE PGSKIP = CNT DEPARTMENT:
DECLARE DATE RPDATE = (*FTODAY*):

ORDER BY DEPARTMENT, EMPLOYEE NUMBER:

SKIP 10 LINES,                                (produces title page)
  PR (26)/MONTHLY SALARIES/:
  PR (26)/     and/:
  PR (26)/   PAY PLAN/:
  SKIP 2 LINES:
  PR (26)/ BY DEPARTMENT/:
FOR PAGE,                                     (date stamps each page)
    AT END,
    PR R(50,MMMBBYY)RPDATE:

FOR DEPARTMENT,
    COMPUTE PGSKIP:
    IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
    PR (11)/Department:/, L(24,X(25))DEPARTMENT:
    PR (11)/-----------/:
    SKIP 2 LINES:
    PR (16)/Employee/,(30)/Monthly/:
    PR (16)/Number/,(30)/Pay Rate/,(45)/Pay Plan/:
    PR (16)/--------/,(30)/---------/,(45)/--------/:

FOR RECORD,
    PR R(16,9999)EMPLOYEE NUMBER,R(30,$ZZZZ.99)PAY RATE,
    L(45,X(7))PAY SCHEDULE:

END REPORT:
```

The picture for PAY RATE replaces the leading zeroes with blanks and prints a '$' on the fifth position.

To generate the report:

```
GENERATE SALARV1 WHERE PAY RATE EXISTS AT 1 AND
DEPARTMENT EXISTS AT 1 AND EMPLOYEE NUMBER LE 1100:
```

```
              MONTHLY SALARIES
                    and
                  PAY PLAN

               BY DEPARTMENT
```

```
Department:  ADMINISTRATION & FINANCE
-----------

      Employee       Monthly
      Number         Pay Rate        Pay Plan
      --------       ---------       --------
      1071 .         $3000.00        MONTHLY
      1086           $   5.85        HOURLY







                                            NOV  91
```

```
Department:  EXECUTIVE
-----------

      Employee       Monthly
      Number         Pay Rate        Pay Plan
      --------       ---------       --------
      1001           $7500.00        MONTHLY
      1002           $4800.00        MONTHLY
      1003           $1500.00        MONTHLY
      1004           $1200.00        MONTHLY
      1005           $3450.00        MONTHLY
      1006           $4500.00        MONTHLY
      1024           $   5.63        HOURLY
      1074           $5250.00        MONTHLY



                                            NOV  91
```

```
Department:  INFORMATION SYSTEMS
-----------

      Employee       Monthly
      Number         Pay Rate        Pay Plan
      --------       ---------       --------
      1008           $3900.00        MONTHLY
      1009           $2250.00        MONTHLY
      1010           $2250.00        MONTHLY
      1012           $3000.00        MONTHLY
      1043           $2100.00        MONTHLY
      1049           $3300.00        MONTHLY
      1067           $4050.00        MONTHLY
      1083           $1350.00        MONTHLY
      1092           $3900.00        MONTHLY


                                            NOV  91
```

```
Department:  MARKETING
-----------

      Employee       Monthly
      Number         Pay Rate        Pay Plan
      --------       ---------       --------
      1007           $4050.00        MONTHLY
      1011           $3000.00        MONTHLY
      1015           $3000.00        MONTHLY
      1017           $   6.38        HOURLY
      1031           $3300.00        MONTHLY
      1050           $3000.00        MONTHLY
      1062           $   6.00        HOURLY
      1077           $2850.00        MONTHLY



                                            NOV  91
```

Notice that the report does not show monthly rates for hourly employees. These rates must be calculated. To arrive at the monthly rate, multiply the hourly rates by 172. To display the correct value for each employee, check each employee record. For hourly employees, print the calculated monthly rate; for monthly employees, print the PAY RATE value.

Here is the next version of the report.

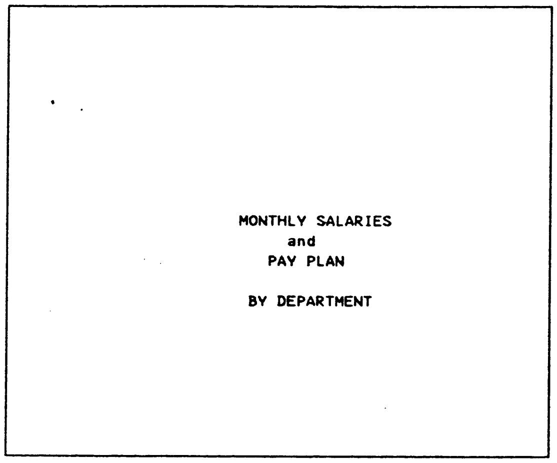
```
REPORT:
FOR REPORT SALARV2:

SUPPRESS TRUNCATION FLAG:

DECLARE PGSKIP = CNT DEPARTMENT:
DECLARE DATE RPDATE = (*FTODAY*):
DECLARE MONEY MNRATE = (PAY RATE * 172.00):   (defines pay rate for
                                                hourly employees)
ORDER BY DEPARTMENT, EMPLOYEE NUMBER:

SKIP 10 LINES,
  PR (26)/MONTHLY SALARIES/:
  PR (26)/     and/:
  PR (26)/   PAY PLAN/:
  SKIP 2 LINES:
  PR (26)/ BY DEPARTMENT/:

FOR PAGE,
    AT END,
    PR R(50,MMMBBYY)RPDATE:

FOR DEPARTMENT,
    COMPUTE PGSKIP:
    IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
    PR (11)/Department:/, L(24,X(25))DEPARTMENT:
    PR (11)/-----------/:
    SKIP 2 LINES:
    PR (16)/Employee/,(30)/Monthly/:
    PR (16)/Number/,(30)/Pay Rate/,(45)/Pay Plan/:
    PR (16)/--------/,(30)/---------/,(45)/--------/:

FOR RECORD,
    IF PAY SCHEDULE EQ HOURLY* THEN
       PR R(16,9999)EMPLOYEE NUMBER,R(30,$ZZZZ.99)MNRATE,
       L(45,X(7))PAY SCHEDULE
    ELSE
       PR R(16,9999)EMPLOYEE NUMBER,R(30,$ZZZZ.99)PAY RATE,
       L(45,X(7))PAY SCHEDULE:

END REPORT:
```

Notice that only one colon is used in the last IF command. If a colon were used after the PRINT command, the REPORT processor would not be able to associate the ELSE with the IF statement.

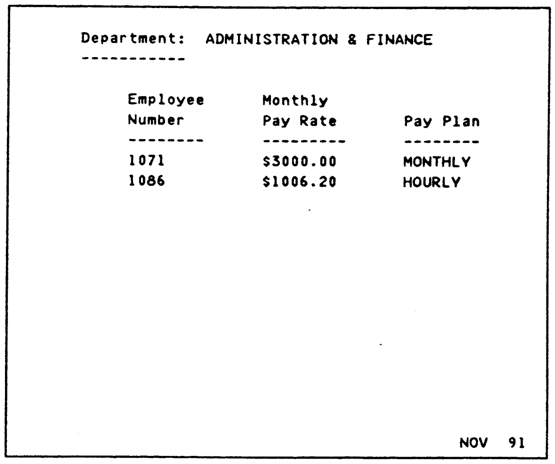
To generate the report:

```
GENERATE SALARV2 WHERE PAY RATE EXISTS AT 1 AND
DEPARTMENT EXISTS AT 1 AND EMPLOYEE NUMBER LE 1100:
```

Notice that the correct value is now printed for employee 1086.



```
MONTHLY SALARIES
      and
   PAY PLAN

 BY DEPARTMENT
```



```
Department:  ADMINISTRATION & FINANCE
-----------

      Employee       Monthly
      Number         Pay Rate       Pay Plan
      --------       ---------      --------
      1071           $3000.00       MONTHLY
      1086           $1006.20       HOURLY




                                              NOV  91
```

The next step prints the total monthly pay of all employees in each department: find the sum of PAY RATE for MONTHLY employees and the sum of MNRATE for HOURLY employees, then add the two totals. To do so, declare three variables:

```
DECLARE MONEY TOTMND = SUM PAY RATE:
DECLARE MONEY TOTHRD = SUM MNRATE:
DECLARE MONEY TOTDPT = (TOTMND + TOTHRD):
```

TOTDPT is an arithmetic expression; therefore, it requires parentheses. Also, to ensure that null values are treated as zeros, you must set the ZERO format option with the QUEST language before starting the REPORT session.

Because these variables use the SUM system function in their declarations, they must be computed as each record containing PAY RATE is processed. Therefore, compute TOTHRD if the record is for an HOURLY employee and compute TOTMND if the record is for MONTHLY employees.

Finally, print the total amount. The total will appear at the bottom of the "Monthly Pay Rate" column, that is, after the records of all employees for a department have been processed. In other words, it appears at the end of the processing for each department.

```
FOR DEPARTMENT,
    .
    .
    .
    AT END,
        PR (32) /---------/:
        PR (15) /Total:/, R(32,$ZZZZZ.99) TOTDPT:
        RESET TOTHRD,TOTMND:
```

TOTMND and TOTHRD are reset (set to null), because the totals are to be calculated anew for the next department.

You must use the RESET command, because the values of TOTMND and TOTHRD are not printed; they are simply used to calculate TOTDPT. If they had been printed, it would have been simpler to declare TOTMND and TOTHRD using the RSUM system function, as RSUM resets after printing the current value.

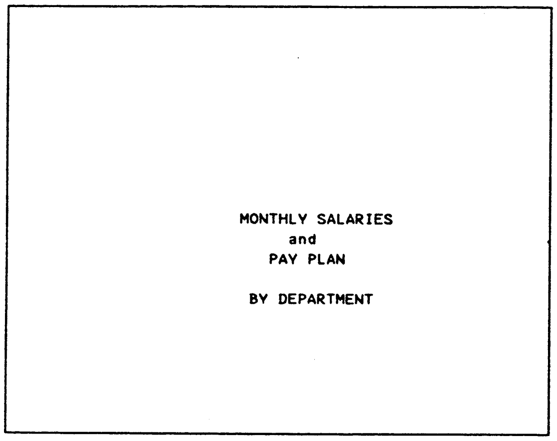
Look at the third version of the report.

```
FORMOP /ZERO/:                              (sets the ZERO format option)
REPORT:
FOR REPORT SALARV3:

SUPPRESS TRUNCATION FLAG:

DECLARE PGSKIP = CNT DEPARTMENT:
DECLARE DATE RPDATE = (*FTODAY*):
DECLARE MONEY MNRATE = (PAY RATE * 172.00):
DECLARE MONEY TOTMND = SUM PAY RATE:
DECLARE MONEY TOTHRD = SUM MNRATE:
DECLARE MONEY TOTDPT = (TOTMND + TOTHRD):

ORDER BY DEPARTMENT, EMPLOYEE NUMBER:

SKIP 10 LINES,
  PR (26)/MONTHLY SALARIES/:
  PR (26)/     and/:
  PR (26)/   PAY PLAN/:
  SKIP 2 LINES:
  PR (26)/ BY DEPARTMENT/:

FOR PAGE,
    AT END,
    PR R(50,MMMBBYY)RPDATE:

FOR DEPARTMENT,
    COMPUTE PGSKIP:
    IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
    PR (11)/Department:/, L(24,X(25))DEPARTMENT:
    PR (11)/-----------/:
    SKIP 2 LINES:
    PR (16)/Employee/,(30)/Monthly/:
    PR (16)/Number/,(30)/Pay Rate/,(45)/Pay Plan/:
    PR (16)/--------/,(30)/---------/,(45)/--------/:

    AT END,
       PR (30) /---------/:
       PR (11) /Total:/, R(30,$ZZZZZ.99) TOTDPT:
       RESET TOTHRD,TOTMND:

FOR RECORD,
    IF PAY SCHEDULE EQ HOURLY* THEN
       COMPUTE TOTHRD,
       PR R(16,9999)EMPLOYEE NUMBER,R(30,$ZZZZ.99)MNRATE,
       L(45,X(7))PAY SCHEDULE
    ELSE
       COMPUTE TOTMND,
       PR R(16,9999)EMPLOYEE NUMBER,R(30,$ZZZZ.99)PAY RATE,
       L(45,X(7))PAY SCHEDULE:

END REPORT:
```

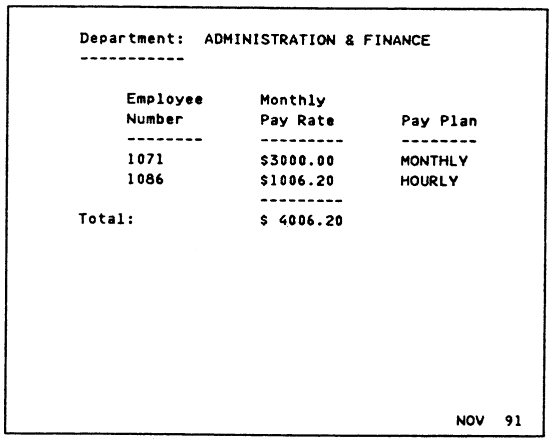
To generate the report:

```
GENERATE SALARV3 WHERE PAY RATE EXISTS AT 1 AND
DEPARTMENT EXISTS AT 1 AND EMPLOYEE NUMBER LE 1100:
```

The first page of the report now shows the total monthly pay for the Administration department.



```
                     MONTHLY SALARIES
                            and
                         PAY PLAN

                       BY DEPARTMENT
```



```
Department:  ADMINISTRATION & FINANCE
-----------

      Employee       Monthly
      Number         Pay Rate        Pay Plan
      --------       ---------       --------
      1071           $3000.00        MONTHLY
      1086           $1006.20        HOURLY
                     ---------
Total:               $ 4006.20




                                              NOV  91
```

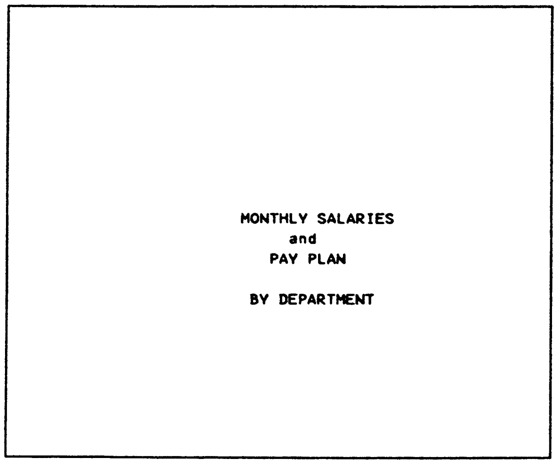
In the next step, print the total number of employees and their average pay for each department. The comments on the next version of the report document the changes.

```
FORMOP /ZERO/:
REPORT:
FOR REPORT SALARV4:

SUPPRESS TRUNCATION FLAG:

DECLARE PGSKIP = CNT DEPARTMENT:
DECLARE DATE RPDATE = (*FTODAY*):
DECLARE MONEY MNRATE = (PAY RATE * 172.00):
DECLARE MONEY TOTMND = SUM PAY RATE:
DECLARE MONEY TOTHRD = SUM MNRATE:
DECLARE MONEY TOTDPT = (TOTMND + TOTHRD):
DECLARE INTEGER EMPCNT = CNT EMPLOYEE NUMBER:     (EMPCNT counts
                                                   number of employees
                                                   in department.)
DECLARE MONEY AVGPAY = (TOTDPT/EMPCNT):           (AVGPAY calculates
                                                   the average pay
                                                   in department.)
ORDER BY DEPARTMENT, EMPLOYEE NUMBER:

SKIP 10 LINES,
  PR (26)/MONTHLY SALARIES/:
  PR (26)/     and/:
  PR (26)/   PAY PLAN/:
  SKIP 2 LINES:
  PR (26)/ BY DEPARTMENT/:

FOR PAGE,
    AT END,
    PR R(50,MMMBBYY)RPDATE:
```

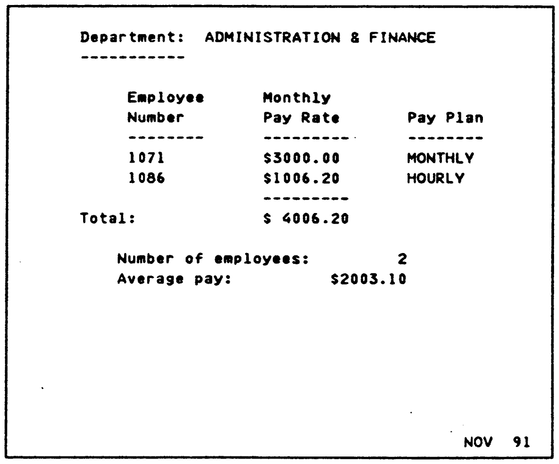
```
FOR DEPARTMENT,
    COMPUTE PGSKIP:
    IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
    PR (11)/Department:/, L(24,X(25))DEPARTMENT:
    PR (11)/-----------/:
    SKIP 2 LINES:
    PR (16)/Employee/,(30)/Monthly/:
    PR (16)/Number/,(30)/Pay Rate/,(45)/Pay Plan/:
    PR (16)/--------/,(30)/---------/,(45)/--------/:

    AT END,
       PR (30) /---------/:
       PR (11) /Total:/, R(30,$ZZZZZ.99) TOTDPT:
                                            (skips 2 lines, print values,)
                                             and resets variables.)
       SKIP 2 LINES:
       PRINT (15)/Number of employees:/,R(43,Z9)EMPCNT:
       PRINT (15)/Average pay:/,R(36,$$$$$$.99)AVGPAY:
       RESET TOTHRD,TOTMND,EMPCNT:

FOR RECORD,
    IF PAY SCHEDULE EQ HOURLY* THEN
                                            (computes number of employees)
       COMPUTE TOTHRD, EMPCNT,
       PR R(16,9999)EMPLOYEE NUMBER,R(30,$ZZZZ.99)MNRATE,
       L(45,X(7))PAY SCHEDULE
    ELSE
                                            (computes number of employees)
       COMPUTE TOTMND, EMPCNT,
       PR R(16,9999)EMPLOYEE NUMBER,R(30,$ZZZZ.99)PAY RATE,
       L(45,X(7))PAY SCHEDULE:

END REPORT:
```

To generate the report:

```
GENERATE SALARV4 WHERE PAY RATE EXISTS AT 1 AND
DEPARTMENT EXISTS AT 1 AND EMPLOYEE NUMBER LE 1100:
```



```
                MONTHLY SALARIES
                      and
                    PAY PLAN

                  BY DEPARTMENT
```



```
Department:  ADMINISTRATION & FINANCE
-----------

     Employee      Monthly
     Number        Pay Rate       Pay Plan
     --------      ---------      --------
     1071          $3000.00       MONTHLY
     1086          $1006.20       HOURLY
                   ---------
Total:             $ 4006.20

    Number of employees:          2
    Average pay:             $2003.10



                                          NOV  91
```

Two more changes finish the report: add summary information at the end, and change some headings and pictures to enhance the format for the entire report.

Declare three new variables: total pay (TOTDPS), total number of employees (TOTEMP), and average pay for all employees (AVGTOT).

Print the three variables at the end of the report by using the AT END statement. (Refer to Page 28 to review how "*n* labels printed" was printed on the previous example.)

Notice that the pictures for MONEY values now include commas.

Look at the fifth version of the report. As before, the code includes comments, and changes are shown in boldface.

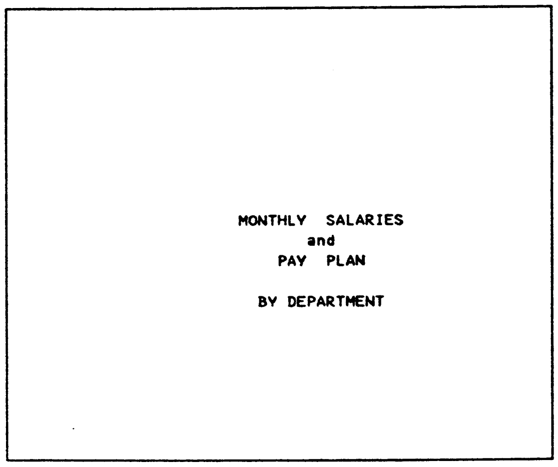
```
FORMOP /ZERO/:
REPORT:
FOR REPORT SALARV5,

SUPPRESS TRUNCATION FLAG:

DECLARE PGSKIP = CNT DEPARTMENT:
DECLARE DATE RPDATE = (*FTODAY*):
DECLARE MONEY MNRATE = (PAY RATE * 172.00):
DECLARE MONEY TOTMND = SUM PAY RATE:
DECLARE MONEY TOTHRD = SUM MNRATE:
DECLARE MONEY TOTDPT = (TOTMND + TOTHRD):
DECLARE MONEY TOTDPS = SUM TOTDPT:            (TOTDPS is the sum of
                                               department pay totals.)
DECLARE INTEGER EMPCNT = CNT EMPLOYEE NUMBER:
DECLARE INTEGER TOTEMP = CNT EMPLOYEE NUMBER: (TOTEMP counts all
                                               employees.)
DECLARE MONEY AVGPAY = (TOTDPT/EMPCNT):
DECLARE MONEY AVGTOT = (TOTDPS/TOTEMP):       (AVGTOT averages totals
                                               for all employees.)
ORDER BY DEPARTMENT, EMPLOYEE NUMBER:

SKIP 10 LINES:
  PR (26)/MONTHLY  SALARIES/:
  PR (26)/       and/:
  PR (26)/    PAY  PLAN/:
  SKIP 2 LINES:
  PR (26)/  BY DEPARTMENT/:

AT END,
   SKIP 10 LINES:
   PR (11)/*****************  SUMMARY  ******************/:
   PR (11)/*/,(55)/*/:
   PR (11)/*/,(55)/*/:
   PR (11)/*/,(14)/Total number of employees:/,R(42,ZZ9)TOTEMP,(55)/*/:
   PR (11)/*/,(55)/*/:
   PR (11)/*/,(15)/Average pay:/,R(42,$$$$,$$$.99)AVGTOT,(55)/*/:
   PR (11)/*/,(55)/*/:
   PR (11)/*/,(15)/Total pay:/,R(42,$$$$,$$$.99)TOTDPS,(55)/*/:
   PR (11)/*/,(55)/*/:
   PR (11)/*/,(55)/*/:
   PR (11)/*********************************************/:
```

```
FOR PAGE,
    AT END,
    PR R(50,MMMBBYY)RPDATE:

FOR DEPARTMENT,
    COMPUTE PGSKIP:
    IF PGSKIP GT 1* THEN SKIP TO NEW PAGE:
    PR (11)/Department:/, L(24,X(25))DEPARTMENT:
    PR (11)/-----------/:
    SKIP 2 LINES:
    PR (16)/Employee/,(30)/Monthly/:
    PR (16)/Number/,(30)/Pay Rate/,(45)/Pay Plan/:
    PR (16)/--------/,(30)/----------/,(45)/--------/:

    AT END,
       COMPUTE TOTDPS:                            (computes total pay.)
       PR (30)/----------/:
       PR (11)/Total:/,R(30,$ZZ,ZZZ.99)TOTDPT:
       SKIP 2 LINES:
       PR (11)/Number of employees:/,R(39,Z9)EMPCNT:
       PR (11)/Average pay:/,R(31,$$$,$$$.99)AVGPAY:
       RESET TOTHRD,TOTMND,EMPCNT:

FOR RECORD,
    IF PAY SCHEDULE EQ HOURLY* THEN
       COMPUTE TOTHRD, EMPCNT, TOTEMP,
       PR R(16,9999)EMPLOYEE NUMBER,R(30,$ZZ,ZZZ.99)MNRATE,
       L(45,X(7))PAY SCHEDULE
    ELSE
       COMPUTE TOTMND, EMPCNT, TOTEMP,
       PR R(16,9999)EMPLOYEE NUMBER,R(30,$ZZ,ZZZ.99)PAY RATE,
       L(45,X(7))PAY SCHEDULE:

END REPORT:
```

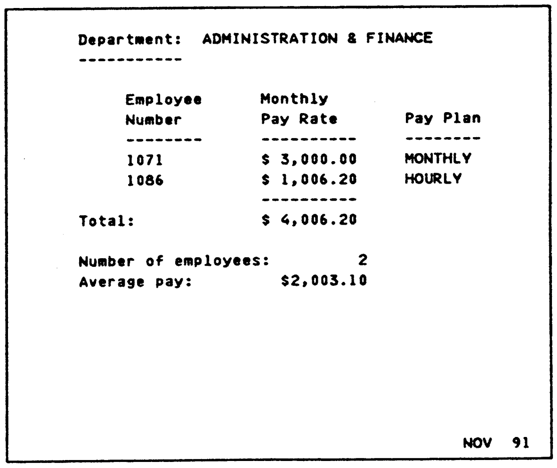
To generate the report:

```
GENERATE SALARV5 WHERE PAY RATE EXISTS AT 1 AND
DEPARTMENT EXISTS AT 1 AND EMPLOYEE NUMBER LE 1100:
```

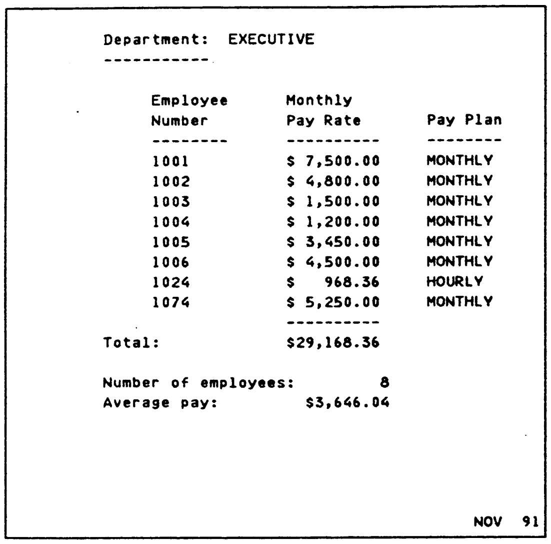
Here is the final version of the report.



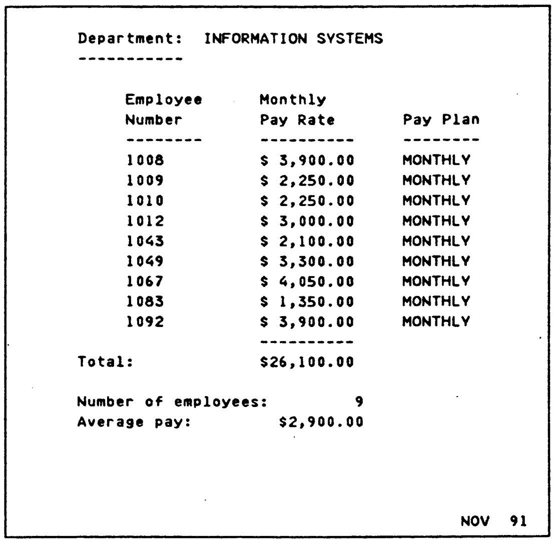
```
MONTHLY  SALARIES
       and
    PAY  PLAN

  BY DEPARTMENT
```



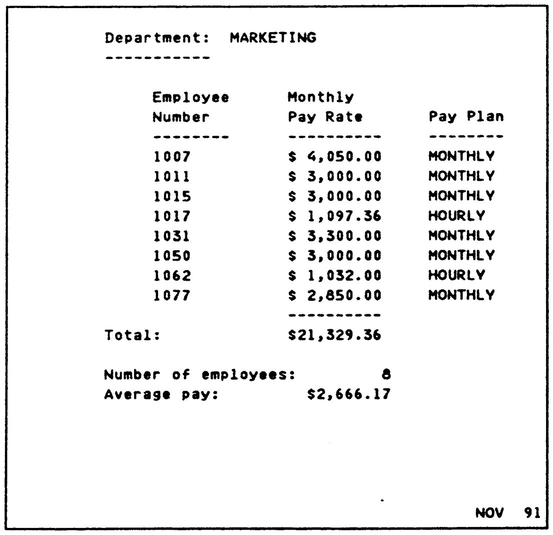
```
Department:  ADMINISTRATION & FINANCE
-----------

    Employee      Monthly
    Number        Pay Rate        Pay Plan
    --------      ----------      --------
    1071          $ 3,000.00      MONTHLY
    1086          $ 1,006.20      HOURLY
                  ----------
Total:            $ 4,006.20

Number of employees:         2
Average pay:          $2,003.10

                                     NOV  91
```



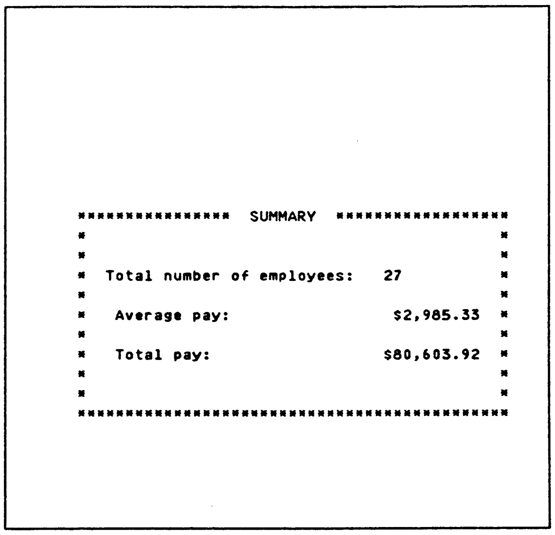
```
Department:  EXECUTIVE
-----------

     Employee      Monthly
     Number        Pay Rate       Pay Plan
     --------      ----------     --------
     1001          $ 7,500.00     MONTHLY
     1002          $ 4,800.00     MONTHLY
     1003          $ 1,500.00     MONTHLY
     1004          $ 1,200.00     MONTHLY
     1005          $ 3,450.00     MONTHLY
     1006          $ 4,500.00     MONTHLY
     1024          $   968.36     HOURLY
     1074          $ 5,250.00     MONTHLY
                   ----------
Total:             $29,168.36

Number of employees:        8
Average pay:         $3,646.04

                                         NOV  91
```



```
Department:  INFORMATION SYSTEMS
-----------

     Employee      Monthly
     Number        Pay Rate       Pay Plan
     --------      ----------     --------
     1008          $ 3,900.00     MONTHLY
     1009          $ 2,250.00     MONTHLY
     1010          $ 2,250.00     MONTHLY
     1012          $ 3,000.00     MONTHLY
     1043          $ 2,100.00     MONTHLY
     1049          $ 3,300.00     MONTHLY
     1067          $ 4,050.00     MONTHLY
     1083          $ 1,350.00     MONTHLY
     1092          $ 3,900.00     MONTHLY
                   ----------
Total:             $26,100.00

Number of employees:        9
Average pay:         $2,900.00

                                         NOV  91
```



```
Department:  MARKETING
-----------

     Employee      Monthly
     Number        Pay Rate       Pay Plan
     --------      ----------     --------
     1007          $ 4,050.00     MONTHLY
     1011          $ 3,000.00     MONTHLY
     1015          $ 3,000.00     MONTHLY
     1017          $ 1,097.36     HOURLY
     1031          $ 3,300.00     MONTHLY
     1050          $ 3,000.00     MONTHLY
     1062          $ 1,032.00     HOURLY
     1077          $ 2,850.00     MONTHLY
                   ----------
Total:             $21,329.36

Number of employees:        8
Average pay:         $2,666.17

                                         NOV  91
```



```
****************  SUMMARY  *****************
*                                          *
*                                          *
*  Total number of employees:   27         *
*                                          *
*   Average pay:               $2,985.33   *
*                                          *
*   Total pay:                $80,603.92   *
*                                          *
*                                          *
********************************************
```

TALLY-WHERE REPORT

This example illustrates how to use the REPORT language to count occurrences of records and values.

For example, the request for the report might be

> *We need a report that tallies the different fields in which employees in the Marketing department majored in college. Print "None" where major field is missing. At the end of the report, list the total number of fields and the total number of occurrences.*

The request cannot be answered with a TALLY command, because the TALLY command does not accept a where-clause, and its output does not display nulls. However, you can use a TALLY command to see what the MAJOR FIELD values look like for all employees. Because MAJOR FIELD is a nonkey item, you must use the CREATE INDEX command first.

```
CONTROL:
CREATE INDEX MAJOR FIELD:
QUEST:
TALLY MAJOR FIELD:
```

```
*******************************************
 ITEM-        MAJOR FIELD
*******************************************
OCCURRENCES   VALUE
-------------------------------------------
        1     ACCOUNTING
        1     ADVERTISING
        1     BIOLOGY
        3     BUSINESS
        3     BUSINESS ADMINISTRATION
        6     COMPUTER SCIENCE
        1     COMPUTER SCIENCES
        1     COMPUTER TECH
        2     COMPUTER TECHNOLOGY
        3     ECONOMICS
        5     EDUCATION
        3     ELECTRICAL ENGINEERING
        2     ENGLISH
        2     FINE ARTS
        1     HISTORY
        1     INFORMATION SYSTEMS
        3     LIBERAL ARTS
        2     MARKETING
        1     MATH
        2     MATHEMATICS
        1     POLITICAL SCIENCE
        1     PSYCHOLOGY
        1     PUBLIC BUSINESS ACCOUNTING
        2     VOCATIONAL OFFICE EDUCATION
-------------------------------------------
       24 DISTINCT VALUES
-------------------------------------------
       49 TOTAL OCCURRENCES
-------------------------------------------
```

The report must print the number of times each MAJOR FIELD value occurs. Therefore, that number must be counted and printed for every value. The report must also count nulls. So, instead of counting each value, count each EDUCATION (C410) record.

In order for the values to be counted correctly, they must be sorted. The REPORT processor counts each record as it processes it. When the value changes, the number of records with that value is printed.

The first version of the report (TALLYV1), is shown below.

```
REPORT:
FOR REPORT TALLYV1:

DECLARE OCCR = CNT C410:

ORDER BY MAJOR FIELD:

FOR MAJOR FIELD,
    AT END,
    IF MAJOR FIELD EXISTS
       THEN PRINT R(5,ZZ9)OCCR, MAJOR FIELD
       ELSE PRINT R(5,ZZ9)OCCR, /None/:
    RESET OCCR:

FOR RECORD,
    AT END,
    COMPUTE OCCR:

END REPORT:
```

To generate the report:

```
GENERATE TALLYV1 WHERE CO HAS C102 EQ MARKETING:
```

```
   1   None
   1   ADVERTISING
   1   BUSINESS
   4   COMPUTER SCIENCE
   1   ECONOMICS
   5   EDUCATION
   2   ELECTRICAL ENGI*
   1   ENGLISH
   1   HISTORY
   2   LIBERAL ARTS
   2   MARKETING
   1   MATHEMATICS
   1   PSYCHOLOGY
   1   PUBLIC BUSINESS*
   1   VOCATIONAL OFFI*
```

The total number of distinct fields and the total number of occurrences are to be printed at the end of the report. To count the distinct fields, set up another variable (DISVAL) to count each time a field is printed. To count the number of values, use another variable (TOTOCC) to count each occurrence. Variable OCCR cannot be used because it is reset after each field is printed.

Because OCCR is reset after printing, the variable can be redefined as an RCNT variable, and the RESET command can be deleted.

Since only the records about college education are pertinent to the report, use a SELECT command to eliminate other education records.

The second version of the report (TALLYV2), is shown below.

```
REPORT:
FOR REPORT TALLYV2:

DECLARE OCCR   = RCNT C410:            (resets value after printing)
DECLARE DISVAL = CNT C410:
DECLARE TOTOCC = CNT C410:

SELECT RECORD IF DEGREE/CERTIFICATE NE HIGH SCHOOL DIPLOMA*:
ORDER BY MAJOR FIELD:

AT END,                                (prints totals at end)
    PRINT R(4,ZZZ9)DISVAL,(12)/Distinct values/:
    PRINT R(4,ZZZ9)TOTOCC,(12)/Total occurrences/:

FOR MAJOR FIELD,
    AT END,
    IF MAJOR FIELD EXISTS
       THEN PRINT R(5,ZZ9)OCCR, L(12,X(30))MAJOR FIELD
       ELSE PRINT R(5,ZZ9)OCCR, (12)/None/:
       COMPUTE DISVAL:                 (counts number of distinct
                                        values)

FOR RECORD,
    AT END,
    COMPUTE OCCR, TOTOCC:              (counts number of occurrences
                                        of value and total)

END REPORT:
```

To generate the report:

```
GENERATE TALLYV2 WHERE CO HAS C102 EQ MARKETING:
```

```
          1    None
          1    ADVERTISING
          1    BUSINESS
          4    COMPUTER SCIENCE
          1    ECONOMICS
          5    EDUCATION
          2    ELECTRICAL ENGINEERING
          1    ENGLISH
          1    HISTORY
          2    LIBERAL ARTS
          2    MARKETING
          1    MATHEMATICS
          1    PSYCHOLOGY
          1    PUBLIC BUSINESS ACCOUNTING
---------------------------------------------------------
         14    Distinct values
         24    Total occurrences
```

The report is basically complete. Two steps remain: to place the totals at the end of the listing (not on the next page) and to add headings.

To place the totals at the end of the listing, you must make sure they appear on the same page as the last value listed. One way is to specify a physical page of infinite length (**PHYSICAL PAGE IS 72 BY 0**). Another way is to print the totals at the end of a FOR block that includes the FOR MAJOR FIELD block. This FOR block must specify an item that has only one value. To do this, define a Collect File item as a constant and then specify the Collect File item in the FOR block.

To use a Collect File item within a report, you must specify that item in the where-clause of the GENERATE command. Since the item has just one value, it can be compared with any other item of similar type.

Headings can be placed at the beginning of the FOR block for the Collect File item.

The third version of the report (TALLYV3) is shown below.

```
COLLECT VAL=(1):                (defines Collect File item as constant)

REPORT:
FOR REPORT TALLYV3:

DECLARE OCCR   = RCNT C410:
DECLARE DISVAL = CNT C410:
DECLARE TOTOCC = CNT C410:

SELECT RECORD IF DEGREE/CERTIFICATE NE HIGH SCHOOL DIPLOMA*:
ORDER BY MAJOR FIELD:

FOR VAL,
    PRINT (4)/Occurrences/,(18)/Value/:
    PRINT (4)/-----------/,(18)/-----------------/:

    AT END,
       PRINT (4)/-----------/,(18)/-----------------/:
       PRINT R(11,ZZZ9)DISVAL,(18)/Distinct values/:
       PRINT R(11,ZZZ9)TOTOCC,(18)/Total occurrences/:

FOR MAJOR FIELD,
    AT END,
    IF MAJOR FIELD EXISTS
       THEN PRINT R(11,ZZZ9)OCCR, L(18,X(30))MAJOR FIELD
       ELSE PRINT R(11,ZZZ9)OCCR, (18)/None/:
       COMPUTE DISVAL:

FOR RECORD,
    AT END,
    COMPUTE OCCR, TOTOCC:

END REPORT:
```

To generate the report:

```
GENERATE TALLYV3 WHERE C0 HAS C102 EQ MARKETING AND C1* NE VAL*:
```

```
Occurrences   Value
-----------   -----------------
          1   None
          1   ADVERTISING
          1   BUSINESS
          4   COMPUTER SCIENCE
          1   ECONOMICS
          5   EDUCATION
          2   ELECTRICAL ENGINEERING
          1   ENGLISH
          1   HISTORY
          2   LIBERAL ARTS
          2   MARKETING
          1   MATHEMATICS
          1   PSYCHOLOGY
          1   PUBLIC BUSINESS ACCOUNTING
-----------   -----------------
         14   Distinct values
         24   Total occurrences
```

SALARY AND SKILLS REPORT

This example illustrates one method for displaying data from different paths in one report. Another method is shown in the next example.

For example, the request for the report might be

> *We need a report that shows the monthly salaries of secretaries and their years of experience in typing and shorthand.*

Start with a LIST command to see the data.

```
LIST PAY RATE, BY ENTRY, SKILL TYPE, YEARS OF EXPERIENCE
WHERE POSITION TITLE EQ SECRETARY AT 0 AND PAY RATE EXISTS AT 0
AND ENTRY HAS (SKILL TYPE EQ TYPING OR SKILL TYPE EQ SHORTHAND):
```

```
*   PAY RATE   SKILL TYPE     YEARS OF EXPERIENCE
***
*  $1,050.00   TYPING                          13
*              ACCOUNTING                       4
*              SHORTHAND                       10
*  $1,237.50   PUBLIC RELAT                     3
               IONS
*              SHORTHAND                        7
*              TYPING                           9
*              ACCOUNTING                       9
*    $900.00   ETS OPERATOR                     1
*              TYPING                           8
*              SHORTHAND                        8
*              ACCOUNTING                       2
*              GERMAN                           3
*      $6.00   GRAPHICS                         3
*              SHORTHAND                        4
*              TYPING                           4
*      $4.88   SHORTHAND                       15
*              TYPING                          19
*      $5.25   TYPING                           6
*              SHORTHAND                        5
*              GRAPHICS                         1
*      $4.50   TYPING                           6
*              SHORTHAND                        5
*              ACCOUNTING                       5
*      $4.88   SHORTHAND                       12
*              ETS OPERATOR                     3
*              TYPING                          23
```

The report, SKILLV1, resembles the Monthly Salaries report. The main difference is that here skill data are also printed. Skill data are in the JOB SKILLS (C200) record; salary data are in the SALARY WITHIN POSITION (C110) record. These records lie in different paths, so the REPORT processor requires that one record be in the primary path and the other have a parent in the primary path.

Use the END PATH WITH command to specify the last record in the primary path:

```
C0 → C100 → C110
```

For each employee, print salary information. Then process each JOB SKILLS record for that employee. If the skill is typing or shorthand, print the number of years of experience.

```
REPORT:

FOR REPORT SKILLV1:

END PATH WITH SALARY WITHIN POSITION:

DECLARE MONEY MNRATE = (PAY RATE * 172.00):    (defines variable for
                                                monthly pay rate)

FOR PAGE,                                      (sets up page headings)
    PRINT (5)/Pay Rate/,  (17)/Typing/, (27)/Shorthand/:
    PRINT (5)/(monthly)/, (17)/(years)/,(27)/ (years)/:
    PRINT (4)/----------/, (17)/-------/, (27)/---------/:

FOR EMPLOYEE NUMBER,                           (prints monthly pay rate)
    IF PAY SCHEDULE EQ HOURLY*
       THEN PRINT R(4,$ZZ,ZZZ.99)MNRATE
       ELSE PRINT R(4,$ZZ,ZZZ.99)PAY RATE:

REPEAT FOR JOB SKILLS:                         (processes Job Skills
                                                records)

       IF SKILL TYPE EQ TYPING*
          THEN SKIP 0 LINES,
               PRINT R(19,Z9)YEARS OF EXPERIENCE:

       IF SKILL TYPE EQ SHORTHAND*
          THEN SKIP 0 LINES,
               PRINT R(30,Z9)YEARS OF EXPERIENCE:

END REPEAT:

END REPORT:
```

To generate the report:

```
GENERATE SKILLV1 WHERE
  POSITION TITLE EQ SECRETARY AT 0 AND PAY RATE EXISTS AT 0 AND
  ENTRY HAS (SKILL TYPE EQ TYPING OR SKILL TYPE EQ SHORTHAND):
```

```
 Pay Rate   Typing   Shorthand
 (monthly)  (years)   (years)
---------   -------  ---------
$ 1,032.00     4         4
$   839.36    19        15
$   903.00     6         5
$   774.00     6         5
$   839.36    23        12
$   900.00     8         8
$ 1,050.00    13        10
$ 1,237.50     9         7
```

TITLES AND MAJORS REPORT

This example shows another method for displaying data from different paths in one report.

For example, the request for the report might be

> *We need a report that shows the pay rates and the major fields of study for employees with Bachelors or Masters degrees.*

In this case, there is no practical way of obtaining the required information with a single LIST command. There is also no way of producing a report, because there are no possible combinations of primary path and secondary record: the path containing record C120 can be used as the primary path, but record C410 is not a child of any record in that path.

The only solution is to collect values in a Collect File, then use them within the report.

Start with the QUEST language COLLECT and LIST commands.

```
COLLECT EMPNUM = C1, MAJOR = C414, DATE = C413, OB C1, C413
WHERE (C412 CONTAINS PREFIX B OR C412 CONTAINS PREFIX M) AND C414 EXISTS:

LIST /TITLE NUMBER, L(30)MAJOR,R(12)DATE/ EMPNUM, MAJOR, DATE:
```

```
* NUMBER   MAJOR                                    DATE
***
*    1002  MATHEMATICS                        06/09/1965
*    1002  BUSINESS ADMINISTRATION            01/07/1969
*    1005  ECONOMICS                          06/12/1971
*    1006  HISTORY                            08/01/1968
*    1006  EDUCATION                          05/30/1973
*    1007  LIBERAL ARTS                       01/11/1984
*    1009  INFORMATION SYSTEMS                01/26/1965
*    1011  MARKETING                          05/21/1977
*    1015  PSYCHOLOGY                         06/01/1980
*    1015  MARKETING                          05/11/1982
*    1031  COMPUTER SCIENCE                   05/01/1980
*    1043  ENGLISH                            05/21/1990
*    1049  FINE ARTS                          08/22/1976
*    1049  COMPUTER TECH                      01/14/1979
*    1050  MATHEMATICS                        06/10/1986
*    1067  MATH                               06/18/1973
*    1067  COMPUTER TECHNOLOGY                01/27/1980
*    1071  BUSINESS ADMINISTRATION            06/07/1975
*    1071  BUSINESS ADMINISTRATION            06/01/1978
*    1077  EDUCATION                          05/29/1975
*    1077  EDUCATION                          06/16/1978
*    1077  EDUCATION                          08/11/1981
           .
           .
           .
*    1161  LIBERAL ARTS                       01/27/1970
*    1217  ADVERTISING                        05/21/1971
*    1234  ECONOMICS                          05/21/1981
*    1238  LIBERAL ARTS                       08/26/1964
*    1238  POLITICAL SCIENCE                  06/22/1966
*    1247  FINE ARTS                          05/24/1988
*    1247  COMPUTER SCIENCES                  07/16/1990
*    1266  ECONOMICS                          05/21/1977
*    1313  COMPUTER SCIENCE                   06/02/1977
*    1313  COMPUTER SCIENCE                   06/18/1986
*    1327  ENGLISH                            01/07/1985
```

Since the report will sort by pay rate, there is no need to sort in the COLLECT command. The report PAYMAJ, which prints the information, is shown below.

The COUNT variable is used to count the MAJOR field and place the field in the appropriate column. The first MAJOR field is printed first, the second one next.

```
COLLECT EMPNUM=C1, MAJOR=C414, DATE=C413 WHERE
(C412 CONTAINS PREFIX M OR C412 CONTAINS PREFIX B) AND C414 EXISTS:

REPORT:
FOR REPORT PAYMAJ,
DECLARE COUNT = CNT OF MAJOR:
ORDER BY PAY RATE, EMPLOYEE NUMBER, DATE:  (keeps data about each
                                            employee together and
                                            sorts by date of degree)

FOR PAGE,
    PR (5)/Pay Rate/,(18)/Major (1st Degree)/,(48)/Major (2nd Degree)/:
    PR (5)/---------/,(18)/------------------/,(48)/------------------/:

FOR EMPLOYEE NUMBER,
    PR R(5,$Z,ZZZ.99)PAY RATE:
    RESET COUNT:                           (resets count of MAJOR
                                            field)

FOR RECORD,
    COMPUTE COUNT:                         (counts MAJOR field)
    IF COUNT EQ 1* THEN SKIP 0, PRINT L(18,X(27))MAJOR:
    IF COUNT EQ 2* THEN SKIP 0, PRINT L(48,X(27))MAJOR:

END REPORT:
```

To generate the report:

```
GENERATE PAYMAJ WHERE (C110 HAS C101 EXISTS AT 1) AT 1
   AND C1* EQ EMPNUM*:
```

```
Pay Rate       Major (1st Degree)             Major (2nd Degree)
---------      ------------------             ------------------
$1,800.00      EDUCATION
$1,800.00      FINE ARTS                      COMPUTER SCIENCES
$1,950.00      ELECTRICAL ENGINEERING
$2,100.00      ENGLISH
$2,250.00      INFORMATION SYSTEMS
$2,250.00      ENGLISH
$2,325.00      LIBERAL ARTS                   POLITICAL SCIENCE
$2,625.00      BIOLOGY
$2,700.00      ECONOMICS
$2,700.00      COMPUTER SCIENCE               COMPUTER SCIENCE
$2,850.00      EDUCATION                      EDUCATION
$2,850.00      ECONOMICS
$3,000.00      MARKETING
$3,000.00      PSYCHOLOGY                     MARKETING
$3,000.00      MATHEMATICS
$3,000.00      BUSINESS ADMINISTRATION        BUSINESS ADMINISTRATION
$3,000.00      PUBLIC BUSINESS ACCOUNTING
$3,000.00      ADVERTISING
$3,150.00      ELECTRICAL ENGINEERING         ELECTRICAL ENGINEERING
$3,300.00      COMPUTER SCIENCE
$3,300.00      FINE ARTS                      COMPUTER TECH
$3,450.00      ECONOMICS
$4,050.00      LIBERAL ARTS
$4,050.00      MATH                           COMPUTER TECHNOLOGY
$4,050.00      LIBERAL ARTS
$4,500.00      HISTORY                        EDUCATION
$4,800.00      MATHEMATICS                    BUSINESS ADMINISTRATION
```

Chapter 4

Executing Reports

This chapter describes how to execute reports, that is, how to actually produce a report after you have defined it.

You can run reports in either single-user or Multi-User environments. In either case, you must have the RW execution parameter set to YES. In single-user jobs, you must define scratch pads. See the *SYSTEM 2000 Product Support Manual, Version 12, First Edition* for details.

Usually, you use your operating system editor to create a file containing your report's definition.

For example, you can create a file to contain the Directory report shown in **Directory Report** on page 14:

```
FOR REPORT DIRECT,
SUPPRESS TRUNCATION FLAG:
    :
    :
END REPORT:
```

To run the report, you could copy that file into your SYSTEM 2000 command stream and submit the job:

```
USER, DEMO:
DBN IS EMPLOYEE:
REPORT:
FOR REPORT DIRECT,
SUPPRESS TRUNCATION FLAG:
    :
    :
END REPORT:
GENERATE DIRECT WHERE C1 EQ 1100 * 1225 AND C102 EXISTS AT 1:
EXIT:
```

An easier way is to make your file into a SYSTEM 2000 Command File, by adding a COMMAND FILE IS command at the end and allocating the file so that it is known to the SYSTEM 2000 session. (See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for information on the Command File.)

```
    :
    :
//DIRECTRY DDNAME ....
    :
    :
FOR REPORT DIRECT,
SUPPRESS TRUNCATION FLAG:
    :
    :
```

[1] In TSO, allocate the file with the ALLOCATE command. In CMS, you would use the FILEDEF command.

```
END REPORT:
COMMAND FILE IS INPUT:
```

To run the report, you can now use the COMMAND FILE IS command from within a SYSTEM 2000 session:

```
USER, DEMO:
DBN IS EMPLOYEE:
REPORT:
COMMAND FILE IS DIRECTRY:
GENERATE DIRECT WHERE C1 EQ 1100 * 1225 AND C102 EXISTS AT 1:
EXIT:
```

Note: If you are in the Multi-User environment, you would use the LOCAL COMMAND FILE IS command to access your own files. (See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for details.)

If you have several reports, you can keep them in separate Command Files and use each one as needed. If one where-clause can be used for several reports, you can generate all of them in one REPORT session:

```
USER, DEMO:
DBN IS EMPLOYEE:
REPORT:
COMMAND FILE IS DIRECTRY:
COMMAND FILE IS ADDRESS:
GENERATE ALL WHERE C1 EQ 1100 * 1225 AND C102 EXISTS AT 1:
EXIT:
```

Reference

Chapter 5

Defining Reports

OVERALL ORGANIZATION

A report definition consists of a collection of blocks, each beginning with a FOR phrase and ending with either the FOR phrase of the next block or with END REPORT (the last block). A block specifies a set of actions and the domain for those actions.

- The FOR REPORT block specifies actions that apply to the report as a whole.
- The FOR PAGE block specifies actions to occur on every output page.
- A FOR *component* block specifies actions to occur at every change in value of the component or Collect File item, or at every new occurrence of a schema record.
- The FOR RECORD block specifies actions to occur for each report record.

A report block can also contain an AT END phrase, specifying actions to occur after the block is processed. The general form of a complete report definition is

```
FOR REPORT report name,
    report-wide specifications
    AT END,
    end-of-report actions

FOR PAGE,
    page actions
    AT END,
    end-of-page actions

FOR component,                    ┐
    breakpoint actions            │  Up to 62 component blocks
    AT END,                       │
    end-of-block actions          ┘

FOR RECORD,
    breakpoint actions
    AT END,
    end-of-block actions

END REPORT:
```

SYNTACTIC CONSIDERATIONS

The actions specified within a block consist of one or more REPORT language commands. A complete command ends with a colon. When a series of consecutive action-clauses, such as PRINT, COMPUTE, and SKIP PAGE, appears in a block, they can be written as separate commands, or they can be separated by commas with a colon after the final clause. For example,

```
FOR PAGE,
    PRINT (5)/Name/,(12)/Phone/:
    PRINT (5)/----/,(12)/-----/:
```

is equivalent to

```
FOR PAGE,
    PRINT (5)/Name/,(12)/Phone/,
    PRINT (5)/----/,(12)/-----/:
```

Commands can be continued on as many lines as needed. You can continue a word from one line to the next as long as all 80 columns are used. The software ignores extraneous blanks between fields and after the system separator.

Although REPORT language commands differ from QUEST language commands, you can invoke both simple and parametric strings in a report definition the same way you invoke them in the QUEST language. This allows you to store any part of a report definition in a database. Strings to be invoked within a report definition must observe the syntax rules defined for REPORT language commands.

Note: If the contents of a string begin with a literal print item, the special characters delimiting the literal must be included within the string definition itself. Otherwise, the string identifier is printed and not its contents.

REPORT RECORDS

The concept of a *report record* and its relationship to the hierarchy of database components is extremely important for accurate report definition.

All database components named anywhere in the report definition must lie along a single vertical path of the database definition unless the disjoint record logic is included in the report definition. See Chapter 9, "Disjoint Data Records in a Report," for details.

A report record is the collection of values corresponding to the named components along a single vertical path of the data. For example, if C1, C105, C113, and C122 are the only database components named in a report definition for the EMPLOYEE database (see foldout), then seven report records could be produced from Logical Entry 2:

```
1120, 03/01/1989, 03/01/1991, 184.00
1120, 03/01/1989, 03/01/1991, 168.00
1120, 03/01/1989, 03/01/1991, 176.00
1120, 03/01/1989, 03/01/1990, 160.00
1120, 03/01/1989, 03/01/1990, 184.00
1120, 03/01/1989, 03/01/1989
1120, 02/16/1988, 02/16/1988
```

Therefore, the potential number of report records from one logical entry is the same as the number of vertical paths that exist between level 0 and the level of the lowest component named anywhere in the report definition.

A report record can also include values for Collect File items. Which value appears in which record depends on how you associate values in the Collect File with the selected records.

Chapter 6

Report-Wide Specifications

INTRODUCTION

Every report definition starts with the report block, which begins with the FOR REPORT phrase and includes commands affecting the entire report. None of the following commands is mandatory. However, any of them that you use must be specified in the order indicated:

```
FOR REPORT phrase
END PATH WITH command
PHYSICAL PAGE IS command
LOGICAL PAGE IS command
INCLUDE/SUPPRESS commands
DECLARE and DECLARE EXTERNAL commands
SELECT command
ORDER BY command
    title page actions
AT END,
    last page actions
```

The commands are described next in the same order.

FOR REPORT PHRASE

The FOR REPORT phrase starts a report definition and names the report. The report's definition ends with the END REPORT command.

Format

```
FOR REPORT name,
```

name is the report identifier. It can be up to 8 alphanumeric characters and must begin with a letter.

Note: The FOR REPORT phrase can end with either a comma or a colon.

END PATH WITH COMMAND

The END PATH WITH command defines the primary path by specifying the lowest schema record in the path. It is required only when a report includes data from disjoint records. (See Chapter 9, "Disjoint Data Records in a Report," for details.)

Format

```
END PATH WITH record:
```

record	is a schema record name or component number.

PHYSICAL PAGE IS COMMAND

The PHYSICAL PAGE IS command defines the dimensions of the report's physical page.

Format

```
PHYSICAL PAGE IS width BY length:
```

width	is the number of print positions per line, up to 132, which is the default.
length	is the number of lines per page. The default is 60. If the length is 0 (zero), the page length is infinite.

Considerations

When you send the report output to a file, column 1 is used for printer carriage control. Since the REPORT processor automatically shifts output one position to the right, a print item specified to begin in column 1 actually begins in the file's second column.

If you specify a length of 0 (zero) and do not subdivide the page into smaller logical pages (using the LOGICAL PAGE IS command), your report consists of one continuous page. Therefore,

- issuing the FOR PAGE phrase and SKIP TO NEW PAGE commands produces error messages.
- issuing SKIP *n* LINES commands is the only way to separate the main portion of the report from the title page or the end page.

The PHYSICAL PAGE IS command is optional. When present, it must be in the position indicated in **Introduction** on page 6-2.

LOGICAL PAGE IS COMMAND

You can subdivide a physical page into smaller logical pages. The LOGICAL PAGE IS command defines the dimensions of the report's logical page.

The software determines the number of logical pages to fit within a physical page by dividing the dimensions of the physical page by those of the logical page.

When you specify the logical page width, you must allow for at least one space between logical pages. For example, if the physical page is 80 by 60, the width of a logical page must be either 80 or less than 30.

You position data on the report according to the logical page dimensions. Multiple logical pages are automatically printed on each physical page. That is, commands that specify pages (the FOR PAGE phrase, the SKIP TO NEW PAGE command) refer to logical pages. For example, consider the following commands:

```
PHYSICAL PAGE IS 100 BY 100:
LOGICAL PAGE IS 30 BY 30:
```

The format and order of the output pages are shown below.

```
Logical                      | |Logical                    | |Logical                     |
Page 1     (30 by 30)        | |Page 2                     | |Page 3                      |
                             | |                           | |                            |
 - - - - - - - - - - - - - - -||- - - - - - - - - - - - - --||- - - - - - - - - - - - - - -|
Logical                      | |Logical                    | |Logical                     |
Page 4                       | |Page 5                     | |Page 5                      |
                             | |                           | |                            |
 - - - - - - - - - - - - - - -||- - - - - - - - - - - - - --||- - - - - - - - - - - - - - -|
Logical                      | |Logical                    | |Logical                     |
Page 7                       | |Page 8                     | |Page 9                      |
                             | |                           | |                            |
 - - - - - - - - - - - - - - -||- - - - - - - - - - - - - --||- - - - - - - - - - - - - - -|
```

In this example, a SKIP 35 LINES command is invalid because it exceeds the logical page length. Similarly, requesting a print item in column 40 would cause an error message.

Format

```
LOGICAL PAGE IS width BY length:
```

width is the number of print positions per line. This number must not exceed the physical page width.

length is the number of lines per page. This number must not exceed the physical page length.

Consideration

The LOGICAL PAGE IS command is optional. When present, it must be in the position indicated in **Introduction** on page 6-2.

INCLUDE/SUPPRESS COMMANDS

The INCLUDE and SUPPRESS commands control the printing of all detail and summary lines of the report, as well as the printing of a truncation flag.

Detail lines and summary lines are arbitrary labels, assigned by you. They enable you to identify certain lines in your report to be either included or suppressed, according to the purpose of the report.

You specify print lines in a report as summary print lines (SPRINT), detail print lines (DPRINT), or simply print lines (PRINT). You control whether detail and summary lines are printed with the INCLUDE/SUPPRESS DETAIL and INCLUDE/SUPPRESS SUMMARY commands, respectively.

Any arithmetic associated with the suppressed lines is still performed; only the printing and any associated SKIP clauses are suppressed (see **SKIP Clause** on page 8-19).

The INCLUDE and SUPPRESS commands are useful if a report is to be run several times, such as a monthly report. By appropriate choice of DPRINT and SPRINT lines in the report definition, you can control how much is printed by changing the INCLUDE and SUPPRESS statement before running the report.

The INCLUDE/SUPPRESS TRUNCATION FLAG command lets you control the overflow flag for |
CHAR, TEXT, and UNDEFINED type values. If a nonblank character is removed by truncation |
at the beginning of the field (right-justified) or at the end of the field (left-justified), the printed value contains the system separator at the beginning or end of the field. If you suppress the truncation flag, the value is still truncated, but the system separator is not printed. (See example **Directory Report** on page 14.)

Format

```
|INCLUDE  DETAIL :
|SUPPRESS
```

```
|INCLUDE  SUMMARY :
|SUPPRESS
```

```
|INCLUDE  TRUNCATION FLAG :
|SUPPRESS
```

The default is INCLUDE.

Abbreviations:
SUPPRESS = SUP
TRUNCATION = TR

Consideration

You can issue the three INCLUDE/SUPPRESS commands in any order, as long as they appear in the position indicated in **Introduction** on page 6-2. If one of the commands is issued multiple times, the setting specified last is in effect for the report generation.

DECLARE COMMAND

A DECLARE command defines a variable for use within the report.

- If you use the DECLARE command, the variable refers to the result of a calculation on the report data. This calculation can be specified as a REPORT language system function (CNT, RCNT, SUM, and RSUM), as an arithmetic expression, or as a stored function.

- If you use the DECLARE EXTERNAL command, the variable refers to the result of a transformation on the report data. This transformation is done by invoking a user-exit. See Appendix A for more details.

The total number of database components and variables that can appear in a report definition is 1000. Of these, a maximum of 500 items can be database components and a maximum of 64 variables can be declared as REPORT language system functions (SUM, RSUM, CNT, RCNT).

If a stored function appears in a DECLARE command, the function must not contain any of the REPORT or QUEST language system functions (AVG, CNT, COUNT, SUM, and so on). All components included in the function must, of course, belong to the same family as the database components named elsewhere in the report definition.

Variable names can be identical with database component numbers or names. For example, the following DECLARE command is valid for the EMPLOYEE database:

```
DECLARE INT C11 = (ACCRUED VACATION + 5):
```

However, if C11 appears elsewhere in the report definition, it is treated as a variable name and not as a component identifier. For example, `FOR C11` would cause an error message, but `FOR ACCRUED VACATION` would not.

The output format of any variable value is determined by the type of edit picture specified in the print clause (see **Print Clauses** on page 8-9).

REPORT Language System Functions

Variables defined with the DECLARE command can use four system functions in their definition: CNT, RCNT, SUM, and RSUM.

CNT and RCNT identify the variable as a counter associated with the named item. Variables defined with RCNT are automatically reset to zero after printing.

SUM and RSUM identify the variable as an accumulator associated with the named item. Variables defined with RSUM are automatically reset to null after printing. SUM and RSUM apply only to numeric (including DATE) items and Collect File items. They will not produce output if any of the summed values are null. To overcome this problem, set the ZERO format option before starting the REPORT session.

Variables that use SUM, RSUM, CNT, and RCNT must be computed by a COMPUTE clause (see **COMPUTE Clause** on page 8-17) before their values can be printed. Other variables refer to the contents of a single report record and are computed automatically.

Variable Types

| Variables can be designated to be type DATE, DOUBLE, DECIMAL, INTEGER, MONEY, or
| REAL.

Variables that are to be treated as type DATE must be specifically designated as such if they are to be printed.

For variables based on calculations on database values (DECLARE command), the type is determined as follows:

- The type specifications are optional.
- | If no specification is given, the result is the highest data type used where high-to-low order is DOUBLE, REAL, DECIMAL, MONEY, INTEGER, and DATE. |
- If the definition uses a stored function, the type is the highest data type used in the function. This result is then converted to the type specified in the function definition and finally converted to the type specified in the DECLARE command. For example, if XYZ is the name of a DECIMAL FUNCTION, then

  ```
  DECLARE INTEGER ABC = *XYZ*:
  ```

  converts the decimal result of the computation to an integer.

Format

```
DECLARE [type] variable = |CNT  [OF] unit :
                          |RCNT
                          |SUM
                          |RSUM
```

```
DECLARE [type] variable = (arithmetic-expression) :
```

```
DECLARE [type] variable = |*function*              :
                          |*function(parameters)
```

type	|DATE |DECIMAL |DOUBLE |INTEGER |MONEY |REAL
variable	is the name of the variable. It can be up to six alphanumeric characters and must begin with a letter. The general considerations given for component names when a component is declared apply to variable names as well. See *SYSTEM 2000 DEFINE Language, Version 12, First Edition* for details. Variables used to define other variables must already be defined.
unit	|*component* |*variable* |(*arithmetic-expression*) |**function** |**function*(*parameters*)
component	|*item* |*cf-item* |*record*
item	is a schema item name or component number.
cf-item	is a Collect File item name. It must be followed by (CF) if it has the same name as a schema component.
record	is a schema record name or component number.
arithmetic-expression	is an expression composed of numeric type items or Collect File items, constants, or previously declared variables, and the operators +, -, *, and /. Date constants must be in MM.DD.YY format or as specified by the DATE FORMAT command. Processing begins with the innermost parentheses and works outward. Then * and / are performed, and finally + and -.
function	is a stored function name or component number.

Considerations

- A variable to be printed in a report becomes an undefined value when it falls outside the range: 5.4×10^{-77} and 7.2×10^{73}.
- DECLARE commands are optional. When present, they must be in the position indicated in **Introduction** on page 6-2.

Examples

These three examples illustrate various ways of defining variables:

- The following variables were defined in **Monthly Salaries Report** on page 31:

```
DECLARE DATE RPDATE = (*FTODAY*):
DECLARE MONEY MNRATE = (PAY RATE * 172.00):
DECLARE MONEY TOTMND = SUM PAY RATE:
DECLARE MONEY TOTHRD = SUM MNRATE:
DECLARE MONEY TOTDPT = (TOTMND + TOTHRD):
```

  Note that the definition of TOTDPT must follow the definitions of TOTMND and TOTHRD.

- The following variables were defined in **Tally-Where Report** on page 42:

```
DECLARE OCCR   = RCNT C410:
DECLARE DISVAL = CNT C410:
DECLARE TOTOCC = CNT C410:
```

  The values of OCCR is reset to zero after every printing. The values of DISVAL and TOTOCC are not reset automatically. You would have to use the RESET command to reset them.

- Assume the following function is added to the EMPLOYEE database definition:

```
1003* YEARS EMPLOYED (DECIMAL FUNCTION $ (*FTODAY* - HIRE DATE)/365) $):
```

  The following DECLARE commands are then valid:

```
DECLARE YEARS = *C1003*:
DECLARE INTEGER YEARS = *C1003*:
DECLARE TOTL = SUM OF *C1003*:
```

  The first DECLARE command simply allows the use of the function elsewhere in the report definition; the second one converts the computed result to an integer.

Abbreviations:

DECLARE	=	DE
DECIMAL	=	DEC
INTEGER	=	INT

SELECT IF COMMAND

The SELECT IF command selects records for the report based on item values within the record or variable values that pertain to the record.

The GENERATE command that causes execution of a report or group of reports can contain a where-clause. The conditions of the where-clause apply to all reports produced as a result of issuing GENERATE. The SELECT IF statement further restricts the records selected for the current report by selecting (from records that satisfy the where-clause of the GENERATE command) those records that also satisfy the conditions of the if-clause.

Format

```
SELECT [RECORD] IF if-clause:
```

if-clause	is a valid REPORT language if-clause as described in **If-Clause** on page 8-4.

Considerations

The if-clause of a SELECT IF command cannot include:

- variables based on a REPORT language system function (CNT, RCNT, SUM, RSUM)
- items from disjoint records (that is, items not in the primary path)
- variables that invoke a user-exit requiring more than one value to produce results.

The SELECT IF command is optional. When present, it must be in the position indicated at the beginning of this chapter.

ORDER BY COMMAND

The ORDER BY command specifies the sort order of records selected for the report.

The order of the sort keys in the ORDER BY command determines the order in which report records are sorted. Specifying a schema record in the ORDER BY causes all report records corresponding to a common schema record occurrence to be grouped together.

If you do not specify an ORDER BY command, the report is generated in random order.

Format

```
ORDER BY sort-keys :
```

sort-keys	is a list of one or more sort keys in the following format, separated by commas:
	`|[LOW]  unit`
	`|[HIGH]`

LOW	specifies low-to-high (ascending) order. The default is LOW.
HIGH	specifies high-to-low (descending) order.

unit	`|item`
	`|cf-item`
	`|record`
	`|variable`

item	is a schema item name or component number.
cf-item	is a Collect File item name. It must be followed by (CF) if it has the same name as a schema component.
record	is a schema record name or component number.
variable	is the name of a variable defined in a DECLARE command.

Considerations

- A maximum of 20 sort keys can be specified in an ORDER BY command.
- A variable cannot be used in an ORDER BY command if it is based on one of the system functions SUM, CNT, RSUM, or RCNT.
- Items from disjoint records (that is, items not in the primary path) cannot be used in an ORDER BY command.

Synonyms:
ORDER BY = ORDERED BY = OB

- Variables that invoke a user-exit requiring more than one value to produce results cannot be used in an ORDER BY command.

- The ORDER BY command is optional. When present, it must be in the position indicated in **Introduction** on page 6-2.

Example

The following example illustrates how specifying a schema record in the ORDER BY command keeps together data about each occurrence of the record.

The left side shows output sorted by DEPARTMENT (C102) values. The right side shows output sorted by DEPARTMENT (C102) and its record, C100. Notice that the output sorted by record keeps data belonging to each C100 occurrence together.

```
ORDER BY C102, C113:
PRINT C1, C102, C113:
```

```
1006   EXECUTIVE          01/01/1990
1006   EXECUTIVE          01/01/1991
1008   INFORMATION S*     01/01/1988
1009   INFORMATION S*     04/01/1988
1008   INFORMATION S*     01/01/1989
1009   INFORMATION S*     09/01/1989
1008   INFORMATION S*     01/01/1990
1009   INFORMATION S*     01/01/1991
1008   INFORMATION S*     01/01/1991
1007   MARKETING          01/01/1988
1007   MARKETING          01/01/1989
1007   MARKETING          01/01/1990
1007   MARKETING          01/01/1991
```

```
ORDER BY C102, C100, C113:
PRINT C1, C102, C113:
```

```
1006   EXECUTIVE          01/01/1990
1006   EXECUTIVE          01/01/1991
1009   INFORMATION S*     04/01/1988
1009   INFORMATION S*     09/01/1989
1009   INFORMATION S*     01/01/1991
1008   INFORMATION S*     01/01/1988
1008   INFORMATION S*     01/01/1989
1008   INFORMATION S*     01/01/1990
1008   INFORMATION S*     01/01/1991
1007   MARKETING          01/01/1988
1007   MARKETING          01/01/1989
1007   MARKETING          01/01/1990
1007   MARKETING          01/01/1991
```

TITLE PAGE ACTIONS

At the end of all the report-wide commands mentioned before, you can specify how the title page of the report is to be printed.

These specifications can include any action-clause (see **Action-Clauses** on page 8-2) as well as IF-THEN-ELSE commands (**If-Clause** on page 8-4).

Any PRINT lines specified in this section of the report precede the body of the report. When either a finite page length (see **PHYSICAL PAGE IS Command** on page 6-5) or logical paging is in effect, these print lines are on a separate page.

If you do not include any specifications, the REPORT processor produces a blank page before printing the next line of the report.

If the PRINT line includes an item or variable, the data printed pertains to the first report record only.

If you include the PAGE system variable (see **Print Clauses** on page 8-9) in the specifications for this page, the variable takes the value 0.

See the sample report in **Monthly Salaries Report** on page 31, for an example of how to produce a title page.

LAST PAGE ACTIONS

The last report-wide specification, the AT END clause, is used to specify how the last (or summary) page of the report is to be printed.

These specifications can include any action-clause (see **Action-Clauses** on page 8-2) as well as IF-THEN-ELSE commands (**If-Clause** on page 8-4).

If you do not include any specifications, the REPORT processor produces a blank page at the end of the report. However, no blank page is produced if you specify an infinite page length and no logical pages (see **PHYSICAL PAGE IS Command** on page 6-5).

If any computing is done in the AT END portion, it pertains only to the values in the last record processed.

See the sample report in **Monthly Salaries Report** on page 31, for an example of how to produce a summary page.

Chapter 7

FOR Blocks

As discussed in **Overall Organization** on page 5-1, a report definition consists of a collection of blocks. A block specifies a set of actions and the domain for those actions.

Remember that each FOR component block is considered to be taking place "within" the preceding FOR component block. Although blocks and their respective AT END phrases are entered sequentially in the report definition, they are nested in their effect on processing report records. For example, consider a report definition with two FOR component blocks and a FOR RECORD block.

```
FOR EMPNUM,                    ┐
   action-clauses              │ Block 1
   AT END,                     │
      action-clauses           ┘

FOR SKILL,                     ┐
   action-clauses              │ Block 2
   AT END,                     │
      action-clauses           ┘

FOR RECORD,                    ┐
   action-clauses              │ Block 3
   AT END,                     │
      action-clauses           ┘
```

Consider this set of ordered report records for EMPNUM and SKILL:

```
1120,COBOL
1120,GRAPHICS
1120,TYPING
1230,TYPING
1230,FORTRAN
1340,COBOL
```

In this situation, the order of processing is as follows:

1. initiate Block 1 actions for EMPNUM = 1120.
2. initiate Block 2 actions for SKILL = COBOL.
3. initiate Block 3 actions for record {1120,COBOL}.
4. perform Block 3 AT END actions for record {1120,COBOL}.
5. perform Block 2 AT END actions.
6. initiate Block 2 actions for SKILL = GRAPHICS.
7. initiate Block 3 actions for record {1120,GRAPHICS}.
8. perform Block 3 AT END actions for record {1120,GRAPHICS}.
9. perform Block 2 AT END actions.
10. initiate Block 2 actions for SKILL = TYPING.
11. initiate Block 3 actions for record {1120,TYPING}.
12. perform Block 3 actions for record {1120,TYPING}.
13. perform Block 2 AT END.
14. perform Block 1 AT END actions.
15. initiate Block 1 actions for EMPNUM = 1230.
16. initiate Block 2 actions for SKILL = TYPING.
17. initiate Block 3 actions for record {1230,TYPING}.
18. and so on.

Three points in particular should be noted in this example:

- A breakpoint on a component or Collect File item causes an automatic breakpoint on subsequent FOR component blocks. For example, in the preceding report definition the FOR EMPNUM block separates record {1120,TYPING} from {1230,TYPING} even though they have the same SKILL value.

- The number of breakpoints depends on the order in which records are sorted by the ORDER BY command and on the sequence of FOR component blocks in the report definition. For example, if the records for the processing example had been ordered on SKILL, FOR EMPNUM would have produced four breakpoints instead of three.

- Inner (lower) block actions are performed completely before the AT END actions of an outer block are performed. If summary actions in the sample FOR EMPNUM block depend on computations performed in the FOR RECORD block, these summary actions should be specified in the AT END portion of the FOR EMPNUM block.

The FOR REPORT block was described in Chapter 6, "Report-Wide Specifications." The FOR PAGE block, the FOR component block, and the FOR RECORD block are described here.

FOR PAGE BLOCK

The FOR PAGE phrase indicates that the actions specified in this block are to be performed for each new page (logical or physical). Only one FOR PAGE block is allowed per report, and it is optional. The FOR PAGE block must appear immediately after the FOR REPORT block.

The print lines specified in this block appear on each new page prior to the print lines specified by any other block. The most common actions in the FOR PAGE block are printing column headings and page numbers, and controlling the vertical spacing between the column heading and the first data line on the page.

If you specify the LOGICAL PAGE IS command (see **LOGICAL PAGE IS Command** on page 6-6), the actions of the FOR PAGE block take place for every new logical page of the report. Otherwise, the actions take place after every page eject, which is automatically performed according to the page length specified in the PHYSICAL PAGE IS command (see **PHYSICAL PAGE IS Command** on page 6-5).

Any printing specified in the AT END portion of the FOR PAGE block is a footing printed as the last information on each page.

Format

```
FOR PAGE,
```

Considerations

- If the PHYSICAL PAGE IS command specifies an infinite page length and there is no LOGICAL PAGE IS command, the FOR PAGE block is invalid.
- The FOR PAGE phrase must end with a comma.

FOR COMPONENT BLOCK

The FOR component blocks indicate that subsequent actions are determined by a change in database values or by a new occurrence of a schema record.

If the component is a database item or a Collect File item, then a breakpoint occurs at every change in the value of that item. That is, the actions specified in the block are performed for each new value of that item. In the example shown below, only four lines will be printed (one for each department, C102), even though there are several different C111 (PAY RATE) values for each department. (To print the various C111 values, you would need a FOR RECORD command.)

```
REPORT:
FOR REPORT TEST2:
    ORDER BY C102:
FOR C102,
    PRINT L(2,X(24))C102, C111:
END REPORT:
GENERATE ALL:
```

In contrast, if you specify a record name or number, the actions in the block are performed at every occurrence of that record. In the example above, if FOR C102 is replaced with C100, one line is printed for every occurrence of the C100 record.

Although report blocks are defined sequentially, each FOR component block can be thought of as occurring "within" the preceding FOR component block. Therefore, the number of times the actions of a FOR component block are performed depends on the order in which report records are sorted, as well as the order of appearance of FOR component blocks.

Format

FOR *component* ,

component	|*item* |*cf-item* |*record*
item	is a schema item name or component number.
cf-item	is a Collect File item name. It must be followed by (CF) if it has the same name as a schema component.
record	is a schema record name or component number.

Consideration

Components from disjoint records (that is, items not in the primary path) cannot be specified in a FOR component phrase.

FOR RECORD BLOCK

In the FOR RECORD block, you specify the actions that are to occur for every report record.

You can specify only one FOR RECORD block in a report definition. If specified, it must be the last block of the report.

Because the FOR RECORD block is always the last one in a report definition, it specifies the "innermost" block of actions performed when the report is executed.

Format

```
FOR RECORD,
```

Consideration

The FOR RECORD phrase must end in a comma.

AT END OF BLOCK

Use the AT END phrase to specify actions to occur at the end of each block.

The statements following the AT END phrase are performed after all other actions for that block of the report are completed. Processing of print lines following the AT END phrase depends on the block type:

- In the FOR REPORT block, any AT END print lines, such as grand totals or final footings, are printed following the body of the report. (See **Last Page Actions** on page 6-17 for details.)
- In the FOR PAGE, any AT END print lines appear as the last information on the bottom of each report page, that is, they specify the page footing.
- In the FOR component and FOR RECORD blocks, any AT END print lines appear at the end of the current block. Such print lines might be section footings, totals, or subtotals.

Format

```
AT END,
```

Consideration

The AT END phrase must end in a comma.

Chapter 8

Specifying and Controlling Actions

ACTION-CLAUSES

REPORT processor action-clauses can be any of the following:

- the print clauses PRINT, DPRINT, and SPRINT
- the arithmetic clauses COMPUTE and RESET
- the CALL clause to pass data to an external subroutine (see Appendix A)
- the SKIP clause.

An action-clause can be specified in three ways:

- by itself, as a separate command
- in a series, with the clauses separated by commas and the entire series terminated by a colon
- in an IF-THEN-ELSE command.

These clauses are available in any report block, but their domain is defined by the block type and by the ordering of records selected for the report.

For example, consider Logical Entry 2 (in the foldout) and assume that the record selection criteria is

```
SELECT RECORD IF C121 OCCURS:
```

A print clause to print EMPLOYEE NUMBER produces different output in the FOR RECORD block than in a FOR C100 block. EMPLOYEE NUMBER values would be printed five times in the FOR RECORD block, but only twice in the FOR C100 block (once for each C100 record). Likewise, the values of variables are dependent on block type. If CNT1=RCNT OF C121, the computed value of CNT1 is always 1 or 0 in a FOR RECORD block because a report record can have only one occurrence of each component, but CNT1 would be 0, 2, or 3 for each C110 record.

LINE and PAGE Variables

The LINE and PAGE variables are integer variables automatically defined by the REPORT processor. You can use these to access the current line number and page number of the report. Whenever LOGICAL PAGE IS has been included in the report definition, these values designate the logical rather than physical line and page numbers. See **Address Labels Report** on page 25 for an example of their use.

IF-THEN-ELSE COMMAND

The IF-THEN-ELSE command chooses between alternate PRINT formats, controls calculations, and determines line spacing.

Format

```
IF if-clause THEN action-clauses ELSE action-clauses :
```

if-clause	is a valid REPORT language if-clause. See **If-Clause** on page 8-4.
action-clause	any number of action-clauses separated by commas. See **Action-Clauses** on page 8-2.

Consideration

The domain of an IF-THEN-ELSE command terminates with the first colon following the THEN or ELSE.

IF-CLAUSE

The if-clause specifies criteria to test report records. You can use it in the IF-THEN-ELSE and SELECT IF commands.

Actions specified after the keyword THEN are performed for those selected data records that satisfy the criteria. If the keyword ELSE is also specified in the command, actions specified after ELSE are performed for those selected data records that do not satisfy the criteria.

Format

```
IF  |if-expression
    |ALL                OF if-expressions
    |ANY [number]
```

```
       if-expression   |if-condition
                       |ALL                OF if-conditions
                       |ANY [number]
```

if-condition can be one of these two forms:

```
|item       OCCURS
|record
```

or

```
|item       |unary-operator
|cf-item    |binary-operator value *
|variable   |* binary-operator item *
            |* binary-operator variable *
```

number	is an integer equal to or greater than one but less than or equal to the number of if-expressions or to the number of if-conditions within the parentheses that follow the ANY OF operator. The default is 1.
item	is a schema item name or component number. The item can be key or non-key.
record	is the record name or component number of a schema record.
cf-item	is a Collect File item name. It must be followed by (CF) if it has the same name as a schema component.
variable	is the name of a variable defined in a DECLARE command.
unary-operator	EXISTS or FAILS.

binary-operator EQ, NE, LE, LT, GE, GT, or the following symbols:

Operator	Symbols
EQ	=
GE	>= or => or ¬< or !<
GT	>
LE	<= or =< or ¬> or !>
LT	<
NE	¬= or !=

value is a literal value or the system string *TODAY*.

Considerations

- Each value specified in an if-clause must have a system separator immediately following it. The default system separator is the asterisk.
- All if-conditions in an if-clause must refer to items belonging to the target record for the command.
- No more than seven if-expressions or if-conditions can be specified within the parentheses following the keyword OF. Therefore, an if-clause cannot include more than 49 if-conditions.
- The threshold operators ANY OF and ALL OF can be nested only once. For example an if-clause of the following form is invalid:

```
IF ANY OF (if-condition1, ALL OF (if-condition2, if-condition3,
   ANY OF (if-condition4, if-condition5)), if-condition6)
```

- The maximum number of characters allowed for any value specified in an if-clause is 250.
- Only items for which you have W-authority can appear in an if-clause.
- Only components from the primary or secondary paths can be specified.
- Without symbols, the software reads a blank on either side of an operator to separate it from other syntax units. However, with symbols, blanks are not required for separation. Any blanks encountered following a symbol are used as part of the value. For example, these two if-clauses are equivalent:

```
IF C1 EQ ABC*
IF C1=ABC*
```

- If component names contain any of the available symbols, use the component numbers. Also, if you change the system separator to be one of the symbols, syntax errors could occur if you specify that symbol in the if-clause.

Abbreviations:

EXISTS = EXIST, EXISTING
FAILS = FAIL, FAILING

Specifying If-Conditions

An if-expression consists of up to seven if-conditions, which you specify with the operators described in this section. Four general types of if-conditions are available. They are described in the next few pages in the order given here:

1. specifying if-conditions that test for the occurrence of components
2. specifying if-conditions that test for the existence of values
3. specifying if-conditions that compare an item, a Collect File item, or a variable with a constant
4. specifying if-conditions that compare an item, a Collect File item, or a variable with another item or variable.

Testing for occurrence You can test for the occurrence of a record or an item in a report record.

A record can occur in a report record even if all its items are null. For example, the following SELECT IF command selects only those report records in which the C120 record occurs (even with null values).

```
   SELECT RECORD IF C120 OCCURS:
```

Testing for an item's occurrence means testing whether the record containing that item exists, even if the item's value is null. For example, the previous command is equivalent to this command:

```
   SELECT RECORD IF C121 OCCURS:
```

Testing for existence You can test whether an item, a Collect File item, or a variable are valued in a report record.

The EXISTS operator tests for the existence of values. EXISTS qualifies all report records that have a value for the specified item.

The FAILS operator tests for the non-existence of values. FAILS qualifies all report records that do not have a value for the specified item.

A variable FAILS if its value is null or undefined, for example, if division by zero has occurred.

A Collect File item exists if a value exists in the Collect File record for the item.

Comparing with a constant The binary operators EQ, NE, LE, LT, GE, and GT (or their equivalent symbols) compare items, Collect File items, or variables with a given value.

The value specified must be compatible with the type of the left-hand operand. For example, a value compared to a type DATE item or variable must be in the form MM/DD/YY or MM/DD/YYYY. If a non-DATE numeric item or variable name appears on the left-hand side, the software converts the specified (numeric) value to match the format of the left-hand operand before the comparison takes place.

EQ qualifies those report records that contain the specified value for the item. For example, this if-condition qualifies those C0 records that contain the value MALE for item C7:

```
IF C7 EQ MALE*
```

NE qualifies those report records containing a value for the item that is not equal to the specified value. For example, the next if-condition qualifies the C0 records that contain a value for C8 that is not equal to CAUCASIAN. It does not qualify those records where the value for C8 is null.

```
IF C8 NE CAUCASIAN*
```

GT (or GE) qualifies those report records containing a value for the item that is greater than (or equal to) the specified value. For example, this if-condition qualifies those C0 records that contain a value for C4 that is greater than or equal to 01/01/91, that is, a date after or on January 1, 1991:

```
IF C4 GE 01/01/91*
```

LT (or LE) qualifies those report records containing a value for the item that is less than (or equal to) the specified value. For example, this if-condition qualifies those C0 records containing a value for C4 that is less than or equal to 01/01/87, that is, a date before or on January 1, 1987:

```
IF C4 LE 01/01/87*
```

The following considerations apply:

- An asterisk (the default system separator) always follows the specified value.
- When a value is specified for a TEXT item, the blank space after the binary operator acts as a delimiter for the operator and is not considered part of the value. However, all blanks after the first and before the asterisk are considered part of the value. For example, this if-condition is satisfied by those selected C130 records that have the value ƀABCƀƀ for item C132:

  ```
  C132 EQƀƀABCƀƀ*
  ```

Comparing two items The binary operators EQ, NE, LE, LT, GE, and GT (or their equivalent symbols) compare items, Collect File items, or variables with another item or variable.

To be eligible for comparison, the items must meet the following requirements:

- The operands on both sides of the operator must be compatible types, either both numeric or both nonnumeric.
- If both operands are numeric, the term on the right-hand side is converted to the exact format of the left-hand term before the comparison takes place.
- The software does not remove any extraneous blanks from a type TEXT item on the right before comparing it to a type CHAR item on the left.

- You can compare a type DATE item or variable with a non-DATE numeric item or variable. The right-hand term is converted to the format of the left-hand term, where the numeric equivalent of a DATE value is the number of days that have elapsed since the advent of the Gregorian calendar.

If either item is null for a particular record, no comparison takes place, and the record cannot satisfy the if-condition.

Note that an asterisk (the default system separator) follows the two items (or variables) being compared.

In the following examples, assume that for the current record ABC is a DECIMAL variable with value 10.1 and C203 is an INTEGER data item with value 10. The record would satisfy the following two if-conditions:

```
IF C203* EQ ABC*
IF C203 EQ 10.1*
```

In both cases, the value (10.1) is converted (truncated) to 10 before the comparison. However, the record would not satisfy the following if-condition because the C203 value (10) is converted to 10.0 before before the comparison:

```
IF ABC* EQ C203*
```

PRINT CLAUSES

A print clause specifies the placement and format of the data and literals for one line. You can define three types of print lines, which means you can define a complex report and produce as many as four different output reports simply by changing the INCLUDE and SUPPRESS commands. (For more details than given here, see **INCLUDE/SUPPRESS Commands** on page 6-7.)

Lines specified with PRINT are always printed.

Lines specified with DPRINT (detail print) are printed by default. To suppress their printing, use the SUPPRESS DETAIL command in the FOR REPORT block.

Lines specified with SPRINT (summary print) are printed by default. To suppress their printing, use the SUPPRESS SUMMARY command in the FOR REPORT block.

All calculations associated with any DPRINT or SPRINT lines are performed even if the line is not printed. For example, consider the following excerpt from a report:

```
       .
       .
       .
DECLARE CNTR=RCNT OF C120:
       .
       .
       .
FOR C110, COMPUTE CNTR,DPRINT R(10,999)CNTR:
       .
       .
       .
```

Since CNTR is defined as the system function RCNT, it is automatically reset after printing. A SUPPRESS DETAIL command would prevent printing CNTR, but the variable would still be reset to zero as if its value had been printed.

Format

```
|PRINT  print-units
|DPRINT
|SPRINT
```

```
         print-unit   |literal-print-unit
                      |data-print-unit
```

These must be separated by commas.

```
 literal-print-unit   / literal /
                      [column] delimiter literal delimiter

    data-print-unit   [ |L (column[,edit-picture]) ] item
                        |R                           cf-item
```

or

```
                      L (column,edit-picture) variable
                      R
```

column	is the first print position of the literal or field.
delimiter	is any special character except the system separator (* by default) or the command terminator (:). The same special character must occur on both sides of the literal. If *column* is omitted, the delimiter must be the slash.
literal	is the actual sequence of characters to be printed. All blanks are preserved in the same manner as for type TEXT data.
L	specifies a left-justified field. Only CHAR, TEXT, or UNDEFINED values can be left-justified.
R	specifies a right-justified field.
item	is a schema item name or component number. The item can be key or non-key.
cf-item	is a Collect File item name. It must be followed by (CF) if it has the same name as a schema component.
variable	is the name of a variable defined in a DECLARE command. The LINE or PAGE variables can also be used.
edit-picture	specifies the format and number of characters in the print field for this item. (See **Printing Data Values** on page 8-11.)

Considerations

- When a print clause contains several print units, you must specify them in order from left to right; that is, in order of increasing column values. They must be separated by commas. If the column specification is omitted, three blank spaces separate the print field from the preceding one.

- A print clause cannot include more than 20 variables that use the RCNT or RSUM functions.

- If a component to be printed is a literal stored in the database as a string, the string definition must include the left-hand special character required as a literal delimiter in the print-unit. Otherwise, enclosing the string reference within special characters in the print-unit itself causes the string name and not its expansion to be printed. For example, consider these two string definitions:

  ```
  1000* ABC (STRING $ ABCDEFGHIJKLMNOPQRSTUVWXYZ $ ):
  1001* ALPHA (STRING /$ ABCDEFGHIJKLMNOPQRSTUVWXYZ $/ ):
  ```

  Then, `PRINT (1) $*ABC*$` produces

  ```
  *ABC*
  ```

  And, `PRINT (1) *ALPHA*` produces

  ```
  ABCDEFGHIJKLMNOPQRSTUVWXYZ
  ```

Printing Data Values

The edit picture specifies the format and number of characters in the print field. If you omit the edit picture, the item's picture designation in the database definition determines the length and format of the print field.

Numeric values are always right-justified. CHAR, TEXT, and UNDEFINED values are printed left-justified, by default.

You can also include insertion or replacement characters in the edit picture of both database components and variables. For example, you can suppress leading zeros in a numeric item, insert hyphens or slashes in a DATE item, or append the symbol DB to a MONEY item.

There are edit pictures for DATE items and for non-DATE items, discussed next.

Edit pictures for DATE values The edit picture for a DATE item specifies the number of characters to be printed for its values.

- Specify MM to print the month as a two-digit number or MMM to print a three-letter abbreviation.
- Specify Y, YY, YYY, or YYYY to print the year.
- Specify DD to represent the day.

You can omit any one or any two of the three parts. If you omit the entire edit picture, the system uses the format MM/DD/YYYY unless you have changed the date format with the QUEST language DATE FORMAT IS command.

You can use any insertion characters such as blanks, commas, or slashes to enhance the readability of DATE values. (See **Edit picture insertion characters** on page 8-13 for details).

The following example illustrates how May 24, 1987, is printed using various edit pictures:

edit picture	output
MMDDYY	052487
DDMMMYY	24MAY87
DDMMY	24057
YYYMMMDD	987MAY24
MMM	MAY
YYYY	1987

Edit pictures for non-DATE values The edit picture for a non-DATE item specifies the type and number of characters to be printed for its values. Specify the type as follows:

A	for alphanumeric values
X	for alphabetic values
9	for numeric values
E	for numeric values

Specify the number of characters either by repeating the A, E, X, or 9, or by indicating the number in parentheses following the A, E, X, or 9. For example, **XXXXX** and **X(5)** produce the same result.

You can use a variety of insertion and replacement characters, such as blanks, commas, or dashes, to enhance the readability of the values. (See **Edit picture insertion characters** on page 8-13 and **Edit picture replacement characters** on page 8-15 for details.)

An edit picture cannot contain more than 30 characters including insertion and replacement characters. Also, the following restrictions apply for numeric values:

format	restrictions
9(*n*)	$1 \le n \le 15$
A(*n*)	$1 \le n \le 15$
9(*n*).9(*m*)	$1 \le n \le 10$ and $1 \le m \le 10$
E(*n*)	$1 \le n \le 15$ (for DECIMAL or INTEGER) $8 \le n \le 14$ (for REAL) $8 \le n \le 24$ (for DOUBLE)

The only restriction for nonnumeric values is that their length must fit in the print line. For example, if the line width is 132 (the default), the following specification would result in an error:

```
L(90,X(50))
```

If the edit picture size is not long enough to print a numeric value, overflow at the left end is truncated and the leftmost character is replaced by the system separator.[1] If overflow occurs at the right end (as for DECIMAL or MONEY values), there is no overflow flag and no rounding.

When a nonnumeric value does not fit into its edit field, its left- or rightmost character is replaced by the system separator. The leftmost character is replaced if the field is right-justified, and the rightmost character is replaced if the field is left-justified. (See **Directory Report** on page 14 for an example.)

To suppress the printing of the overflow flag, use the SUPPRESS TRUNCATION FLAG command in the FOR REPORT block. (See **INCLUDE/SUPPRESS Commands** on page 6-7.)

If a variable value is undefined (for example, if division by zero occurred), the print item is printed as Xs the entire width of the edit picture. Wherever a variable value requires computation using a null, the result is null and no printing is performed unless you have set the ZERO format option (see **Monthly Salaries Report** on page 31). Similarly, if the print item is a database item, nulls are not printed.

Unprintable characters in UNDEFINED values are replaced with a period.

[1] The asterisk (*) is the default system separator. You can change it with the SEPARATOR IS command. See *SYSTEM 2000 CONTROL Language, Version 12, First Edition.*

The following examples are valid edit pictures for non-DATE values:

source data	edit-picture
1120	9(4)
12	99
INFORMATION SYSTEMS	A(20)
441-04-0121	X(11)

Edit picture insertion characters You can enhance the readability of output values by including the following insertion characters in the edit picture of the item:

```
$ - + . , 0 B < > / CR DB
```

In addition, the following insertion characters are available for CHAR, DATE, TEXT, and UNDEFINED items:

Z	floating +	floating <
*	floating -	floating $

where "floating" means multiple consecutive occurrences of the character.

Insertion characters appear in the output data item, so they must be counted in the picture size to avoid overlapping print fields. The usage of each character is as follows:

$ When you specify a single dollar sign in a numeric item edit picture, it must appear as the leftmost character of the edit picture unless preceded by +, -, or <. The dollar sign can appear anywhere in the edit picture of a nonnumeric item.

+ A plus sign in a numeric item edit picture must be either the rightmost symbol or the leftmost symbol unless preceded by a dollar sign. A plus sign is inserted in the corresponding character position of the edited value if the value is positive or unsigned. If the value is negative, a minus sign is inserted.

- A single minus sign in a numeric item edit picture must be either the rightmost symbol or the leftmost symbol unless preceded by a dollar sign. If the value is negative, a minus sign is inserted in the corresponding character position of the printed value. If the value is not negative, a blank is inserted in the designated position; it is counted in the size of the item. A minus sign specified in any position other than first or last is inserted in the position indicated.

. The period represents an actual decimal point as opposed to an assumed decimal point. When specified, a decimal point appears in the printed value as a character in the corresponding character position. A numeric item edit picture can contain only one decimal point; a nonnumeric or DATE item may have more.

, When you specify a comma, it is inserted in the corresponding character position of the printed value.

0 When you specify a zero, it is inserted in the corresponding character position in the printed value.

B When you specify a B, a blank is inserted in the corresponding character position of the printed value.

CR You can specify the credit symbol CR only for numeric items. The CR must be placed at the right end of the edit picture. If the value is negative, the CR is inserted in the last two character positions of the printed data item. If the value is positive or unsigned, these last two character positions are set to blanks. Thus, this symbol always occupies two characters (CR or spaces) in the size of the item.

DB You can specify the debit symbol DB only for numeric items. The DB must be placed only at the right end of the edit picture. It functions for positive values in the same manner as the credit symbol functions for negative values.

/ When you specify a slash, it is inserted in the corresponding character position in the printed value.

< You can specify the left angle bracket only for numeric items. The < must be placed only at the left end of the edit picture. It functions in the same manner as the credit symbol.

> You can specify the right angle bracket only for numeric items. The > must be placed only at the right end of the edit picture. It functions in the same manner as the credit symbol.

Examples

Source Data	Edit Picture	Printed Output
48.34	$99.99	$48.34
4834	9,999	4,834
292	+999	+292
-292	+999	-292
292	-999	292
-292	-999	-292
345	<999>	345
-345	<999>	(345)
234.56	$BB999.99	$ 234.56
234.56	$00999.99	$00234.56
-1.2E+25	E(8)	-1.2E+25
1.2E+25	E(10)	1.200E+25
1.234E-25	E(10)	1.234E-25
1.234E+25	E(14)	1.2339999E+25
June 5, 1985	MM/DD/YY	06/05/85
	MMMBDD,YYYY	JUN 05,1985
	DD-MM-YY	05-06-85
	YYYY--MMMBDD	1985--JUN 05
ABCDE	XXXXX	ABCDE
ABCDE	XX,XXBX	AB,CD E
ABCDE	XX-X-XX	AB-C-DE
ABCDE	.X0XXX0X.	.A0BCD0E.
X'01C101C201'	X(5)	.A.B.
X'01C101C201'	XX,XXBX	.A.B.

The following examples illustrate truncation of data with insertion characters when overflow would occur. The asterisk (*) is the system separator.

Database Picture	Source Data	Edit Picture	Left-justified Output	Right-justified Output
X(4)	ABCD	/XXX/	/ABC/	/BCD/
X(5)	PQRST	XX-XX	PQ-RS	QR-ST
X	W	XX-XX	W -	- W
XXXX	WYƀƀ	-XX-	-WY-	- -
9(4).9(2)	1234.56	$00999.99	*0 234.56[2]	*0234.56

Note that trailing blanks will occur only in TEXT values in the database.

Edit picture replacement characters Replacement characters in the edit pictures of DECIMAL, INTEGER, and MONEY data suppress leading zeros and replace them with other characters in output. Only one type of replacement character can be used in an edit picture, although pictures with replacement characters can also contain insertion characters according to the rules in the following discussion. Replacement characters are

Z	floating +	floating <
*	floating -	floating $

where "floating" means multiple occurrences of the character. These characters are insertion characters in nonnumeric values.

Z You can specify a Z at the left end of the edit picture once for each leading zero that is to be replaced by blanks in the printed value. Zs may be preceded by one of the insertion characters [, $, +, or -, and interspersed with any of the period, comma, zero, slash, minus sign, or B insertion characters.

Only the leading zeros that occupy a position specified by Z are suppressed and replaced with blanks. Zeros are never suppressed to the right of the first nonzero digit, nor are any zeros suppressed to the right of a decimal point unless the value of the data is zero and all the character positions in the item are described by a Z. In this case, the edited item consists of all blanks and even the decimal point is suppressed.

If <, $, +, or - precedes the Zs, it is inserted in the far left character position of the value regardless of any leading zeros suppressed in the value. When the value is zero and all the following character positions are specified by Zs, the <, $, +, or - is replaced by a blank (- is treated as a minus sign).

If a 0, comma, -, B, or / in the edit picture is encountered before zero suppression terminates, the character is suppressed and a blank inserted in its place (- is treated as a minus sign).

* The asterisk replaces a leading zero in the corresponding character position with an asterisk instead of a blank. It is specified like the editing character Z and follows the same rules except that it never replaces a decimal point.

[2] If you specify left justification of numeric data, a warning message is issued and the output values are right-justified.

$ When you specify a dollar sign as a replacement character to suppress leading zeros, it acts as a floating dollar sign and is printed directly preceding the first nonsuppressed character. One more dollar sign must be specified than the number of zeros to be suppressed. This dollar sign is always printed in the edited data whether or not any zero suppression occurs. The remaining dollar signs act in the same way as Z to suppress leading zeros. Only <, +, and - may precede the initial dollar sign as insertion characters. Each dollar sign specified in an edit picture is counted in determining the size of the report item.

+ When you specify a plus sign as a replacement character, it is a floating plus sign and is specified one more time than the number of leading zeros to be suppressed. It functions in the same way as the floating dollar sign: a plus sign is placed directly preceding the first nonsuppressed character if the value is positive or unsigned, and a minus sign is placed in this position if the value is negative. A dollar sign can precede a floating plus sign as an insertion character.

- When you specify a minus sign as a replacement character, it is a floating minus sign and is specified one more time than the number of leading zeros to be suppressed. It functions in the same way as the floating plus sign except that a blank is placed directly preceding the first nonsuppressed character if the value is nonnegative. A dollar sign can precede a floating minus sign.

< When you specify a left angle bracket as a replacement character, it is a floating left angle bracket and functions like a floating minus sign. The left angle bracket is specified one more time than the number of leading zeros to be suppressed. A dollar sign may precede a floating left angle bracket.

Examples

Source Data	Edit Picture	Printed Output
1234	ZZZ9	1234
0234	ZZZ9	234
00.00	ZZ.Z9	.00
00.00	ZZ.ZZ	
0012.34	$****.99	$**12.34
0012.34	$$$$$.99	$12.34
001234.	$***,**9.99	$**1,234.00
0012.34	<$$$$$.99>	$12.34
-0012.34	<$$$$$.99>	($12.34)
-0012.34$	<<<<<.99>	$ (12.34)

COMPUTE CLAUSE

The COMPUTE clause evaluates one or more variables previously defined in a DECLARE command as a system function.

Use the COMPUTE clause to generate the current value of a system function before the system function or a variable dependent on it is printed. COMPUTE has no effect on variables that are not defined as system functions.

Format

```
COMPUTE variables
```

variable	is a variable defined in a DECLARE command with the CNT, SUM, RCNT, or RSUM system function.

See the examples of COMPUTE in **Monthly Salaries Report** on page 31.

Abbreviation:

COMPUTE = CO

RESET CLAUSE

The RESET clause resets the values of one or more variables defined as system functions. CNT and RCNT are reset to zero; SUM and RSUM are set to null.

To reset the value of a variable defined as the system functions SUM or CNT, you must use the RESET clause. You can also use the RESET clause for RSUM and RCNT variables, since these are reset automatically only when they are printed.

See **Monthly Salaries Report** on page 31, **Tally-Where Report** on page 42, and **Titles and Majors Report** on page 49 for examples of the RESET clause.

Format

```
RESET variables
```

variable	is a variable defined in a DECLARE command with the CNT, SUM, RCNT, or RSUM system function.

Considerations

- You cannot reset variables that are not system functions. Their values are always based on the contents of a single record only.
- The PAGE and LINE integer variables can be reset to zero. However, be careful when resetting LINE and PAGE in conjunction with SKIP clauses to prevent undesirable vertical placement of print lines.

Abbreviation:
RESET = RS

SKIP CLAUSE

The SKIP clause controls vertical placement of report lines on a page.

There is an implicit SKIP 1 LINE assumed before every print clause. You can override this action by preceding the PRINT clause with a SKIP clause. SKIPs are always associated with the next PRINT clause unless a FOR, THEN, ELSE, or AT END unit intervenes. If you specify a DPRINT or SPRINT, the SUPPRESS DETAIL or SUPPRESS SUMMARY suppresses the SKIPs as well as the printing.

Format

```
SKIP [TO NEW] PAGE
SKIP n [LINE[S]]
```

n is the number of lines to be skipped; $0 \le n \le 63$.

SKIP PAGE causes a page eject, sets the variable LINE to 1, and increments variable PAGE by 1.

SKIP *n* also causes a page eject if skipping *n* lines would result in a line number greater than the maximum page length. If the report definition includes the LOGICAL PAGE IS command, the SKIP clauses refer to logical page size. Whenever a page eject occurs, all FOR PAGE computations and print lines are produced as usual at the end of the current page and the top of the new page.

If $n = 0$, overprinting results. However, overprinting is not done with carriage control characters. The data printed after SKIP 0 replace any data already printed on the line.

For example, for FULL TIME employee GIBSON, these lines

```
PRINT EMPLOYEE STATUS
SKIP 0:
PRINT LAST NAME:
```

would produce

```
GIBSONIME
```

Considerations

- If you specify an infinite page length in the PHYSICAL PAGE IS command and you have not issued a LOGICAL PAGE IS command, SKIP TO NEW PAGE is invalid and produces an error message.
- The SKIP PAGE clause is not allowed in any of the following blocks: FOR REPORT block, AT END of the FOR REPORT block, FOR PAGE block, and AT END of the FOR PAGE block.

Examples

The following statement results in double spacing because SKIP 2 LINES overrides the implicit SKIP preceding PRINT:

```
FOR RECORD,SKIP 2 LINES,PRINT... :
```

In the following statement, the SKIP clause is again associated with the print clause, even though a COMPUTE clause intervenes. If SUPPRESS SUMMARY were in effect, the SPRINT and SKIP actions would both be suppressed and the COMPUTE would be performed just as if the SPRINT items were to be printed.

```
... SKIP TO NEW PAGE,COMPUTE TOTL,SPRINT... :
```

Chapter 9

Disjoint Data Records in a Report

A report generated by the REPORT processor usually includes data from only a single path of the database definition. However, a report can also include data from items belonging to any schema record whose parent lies in that path.

By default, the REPORT processor looks for the lowest-level record in your report definition. The processor then creates the report records (see **Report Records** on page 5-3) by selecting data from that record and its ancestors.

If your report is to include data from more than one path, you must use the END PATH WITH command to specify the primary path for your report. The primary path consists of the lowest-level record in your report and its ancestors.

As mentioned above, you can include data from records whose parent lies in the primary path. These records, which are disjoint from records in the primary path, are called secondary records. For example, if the report includes this command that refers to the C110 record in the EMPLOYEE database

```
END PATH WITH C110:
```

the primary path is

```
C0 → C100 → C110
```

The report can include data from any of these secondary records: C200, C300, C400, and C130, but the report cannot include data from C410 or C420.

By default, if your report includes an item from a secondary record, only the value from the first occurrence of that record is printed each time a value from the parent record is printed.

To print values for the other occurrences, you need to include a REPEAT FOR / END REPEAT block. To skip an occurrence within a BLOCK, use the CYCLE action.

The rest of this chapter describes the syntax for the commands mentioned above and presents a detailed example. See **END PATH WITH Command** on page 6-4 for the syntax of the END PATH WITH command.

REPEAT BLOCKS

The REPEAT FOR command marks the beginning of a REPEAT block. The REPEAT block lets you access values in multiple occurrences of secondary records.

For each occurrence of the report record, a REPEAT FOR block is executed once for each occurrence of the records specified in the REPEAT FOR command. The number of iterations is determined by the record with the greatest number of occurrences. Nulls are not ignored; that is, an occurrence of the record triggers an iteration of the REPEAT block whether the items referenced within the block have values or not. At least one of the records specified in the REPEAT FOR command must occur for the REPEAT block to be executed.

For example, assume that a REPEAT FOR command specifies two schema records: SR1 and SR2. If, for an occurrence of the lowest record in the primary path (the end-path record), SR1 occurs three times and SR2 occurs four times, the REPEAT block is executed four times. In the last iteration, nulls are used for the SR1 values. (See example.)

Any reference within a REPEAT block to items in the records specified in the REPEAT FOR command pertains to the item's current value.

The END REPEAT command marks the end of a REPEAT block. When the REPORT processor reaches the END REPEAT command, it checks for more occurrences of all items in secondary records referenced in the REPEAT block. If any new values remain for any secondary record item specified in the REPEAT block whose record is also specified in the REPEAT FOR command, the processor sets pointers to the next occurrence and executes the block again. If no new values remain, the execution of the REPEAT block is terminated and the processor sets pointers to the last value of the item for each secondary record.

Format

```
REPEAT FOR records:
END REPEAT:
```

records	is the record name or component number of a secondary record. No more than 100 records can be specified, separated by commas.

Considerations

- You cannot order the values within a REPEAT block. Values are accessed in logical order.
- A FOR block or an AT END block cannot be specified within a REPEAT block. Also a REPEAT block cannot span a FOR block or an AT END block.
- REPEAT blocks cannot be nested. If the processor finds a REPEAT FOR command within a REPEAT block, it assumes an END REPEAT command and ends the first block.
- Items from secondary records referenced outside a REPEAT block are treated according to the following rules:

a) The processor uses the value of the first occurrence within the current record if the item is referenced before the execution of the first REPEAT block that both uses the item within the block and specifies the item's record in the REPEAT FOR command. (See **Example** on page 9-4.)

b) The processor uses the value of the last occurrence within the current record if the item is referenced after the execution of a REPEAT block that both uses the item within the block and specifies the item's record in the REPEAT FOR command. (See **Example** on page 9-4.)

CYCLE ACTION

Use the CYCLE action to bypass actions in a REPEAT block when certain conditions are true. In effect, the CYCLE action is equivalent to a "go to END REPEAT"; it allows you to bypass a specific iteration.

Format

```
CYCLE
```

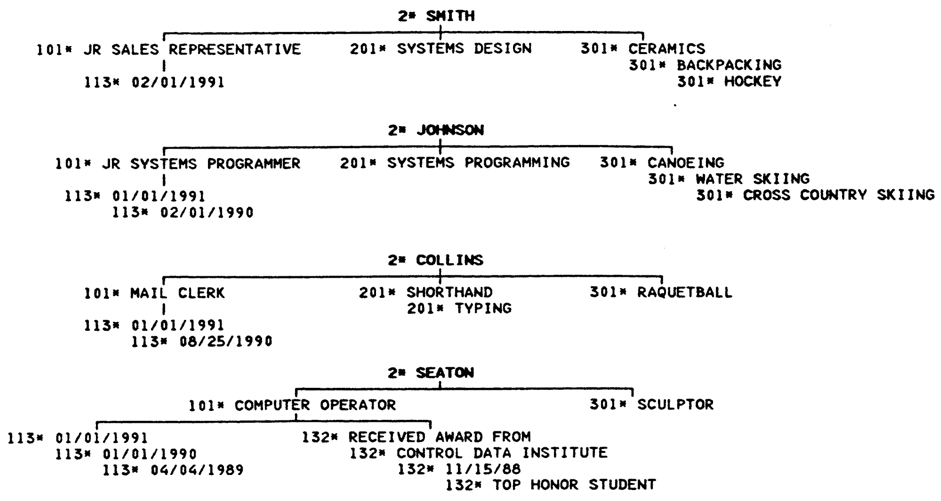
Considerations

- The CYCLE action can be used only in a REPEAT block.
- The CYCLE action can be used only in an IF-THEN-ELSE command. (See example.)
- No more than 100 CYCLE actions can be given in one REPEAT block.

EXAMPLE

The following example illustrates how the REPORT processor prints data from disjoint records. The example includes the REPORT commands that generate the report, the data from the four logical entries used in the report, and the actual report. To enhance readability, the output data appear in boldface.

```
REPORT:
FOR REPORT DISJ,
    END PATH WITH C110:
    DECLARE NUMSK = RCNT OF C201:

FOR C2,
    SKIP 2 LINES:
    PRINT /C2: /, L(6,X(10)) C2:

      REPEAT block is executed once for each occurrence of C300.
       At each iteration, a new C301 value is printed.  Only the first
       C201 value is printed, since C200 is not specified in the
       REPEAT FOR command.  The CYCLE action causes the BACKPACKING
       record to be skipped.

    REPEAT FOR C300:
      IF C301 EQ BACKPACKING* THEN CYCLE:
      PRINT (3) /C301: /, L(10,X(20)) C301, /C201: /,L(40,X(20)) C201:
    END REPEAT:

      The first occurrence of C201 is printed because C200 was
       not specified in the previous REPEAT FOR command.

    PRINT (6) /C201: /, L(13,X(20)) C201:

    REPEAT FOR C200:
        COMPUTE NUMSK:
    END REPEAT:

      The last occurrence of C201 is printed because C200 was
       specified in the previous REPEAT FOR command and C201 was used
       in that REPEAT block to compute NUMSK.

    PRINT (6) /SUMSK: /, R(14,99) NUMSK, /C201: /, L(26,X(20)) C201:

 FOR C101:

      The first occurrences of both C101 and C132 are printed.

    PRINT (3) /C101: /, L(10,X(25)) C101, /C132: /, L(45,X(25)) C132:
    REPEAT FOR C130, C300:

      All occurrences of C132 and C301 are printed because both C130
       and C300 appear in the REPEAT FOR command.  The number of
       repetitions is determined by the record having the most
       occurrences.

       PRINT (6) /C132: /, L(13,X(25)) C132, /C301: /, L(48,X(20)) C301:
    END REPEAT:

FOR C113,
    PRINT (3) /C113: /, R(10,MMMYY) C113:
END REPORT:
```



```
C2:  SMITH
  C301:  CERAMICS              C201:  SYSTEMS DESIGN
  C301:  HOCKEY                C201:  SYSTEMS DESIGN
     C201:  SYSTEMS DESIGN
     SUMSK:  01  C201:  SYSTEMS DESIGN
  C101:  JR SALES REPRESENTATIVE    C132:
     C132:                          C301:  CERAMICS
     C132:                          C301:  BACKPACKING
     C132:                          C301:  HOCKEY
  C113:  FEB91

C2:  JOHNSON
  C301:  CANOEING              C201:  SYSTEMS PROGRAMMING
  C301:  WATER SKIING          C201:  SYSTEMS PROGRAMMING
  C301:  CROSS COUNTRY SKIING  C201:  SYSTEMS PROGRAMMING
     C201:  SYSTEMS PROGRAMMING
     SUMSK:  01  C201:  SYSTEMS PROGRAMMING
  C101:  JR SYSTEMS PROGRAMMER      C132:
     C132:                          C301:  CANOEING
     C132:                          C301:  WATER SKIING
     C132:                          C301:  CROSS COUNTRY SKIING
  C113:  JAN91
  C113:  FEB90

C2:  COLLINS
  C301:  RAQUETBALL            C201:  SHORTHAND
     C201:  SHORTHAND
     SUMSK:  02  C201:  TYPING
  C101:  MAIL CLERK                 C132:
     C132:                          C301:  RAQUETBALL
  C113:  JAN91
  C113:  AUG90

C2:  SEATON
  C301:  SCULPTOR              C201:
  C301:  HIKING                C201:
     C201:
     SUMSK:  00  C201:
  C101:  COMPUTER OPERATOR          C132:  RECEIVED AWARD FROM
     C132:  RECEIVED AWARD FROM     C301:  SCULPTOR
     C132:  CONTROL DATA INSTITUTE  C301:  HIKING
     C132:  11/15/88                C301:
     C132:  TOP HONOR STUDENT       C301:
  C113:  JAN91
  C113:  JAN90
  C113:  APR89
```

Chapter 10

The GENERATE Command

After you have completed defining the report (or reports), you execute it (or selected reports) with the GENERATE command. In effect, the GENERATE command marks the end of a REPORT session. If you are planning to include a where-clause in a GENERATE command, any reports you define in the REPORT session must be generated from report records that satisfy the conditions of that where-clause.

The GENERATE command allows you to generate as many as 100 formatted reports from a single scan of a database.

Format

```
GENERATE | reports  [where-clause]:
         | ALL
```

report	is the name assigned to the report in the FOR REPORT phrase.
ALL	indicates that all reports defined in the REPORT session are to be generated.
where-clause	is a QUEST language where-clause. See *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition* for details. In some cases, described later, the where-clause is required.

Once the GENERATE command executes successfully, you can no longer refer to those reports. If you are working interactively, the report definitions are lost unless you have saved them ahead of time or you have made provisions to save the Message File.

Because report definitions are not saved after GENERATE has successfully completed, you will most often define reports in groups and generate all of them by specifying ALL in the GENERATE command. The reports are generated in the order you defined them. However, if any of those report definitions contains errors, the corresponding reports are not generated. You must redefine these reports in a new REPORT session.

If you specify individual report names in the GENERATE command, these are executed in the order they appear in the command rather than in the order they were defined. If any of the named reports contains an error, execution of the GENERATE command is prevented, although this is not the case when ALL is specified.

Where-Clause Processing

You must specify a where-clause in the GENERATE command if the report definition includes both database items and Collect File items. In all other cases, a where-clause is optional.

If you specify a where-clause, consider the following:

- If the report definition includes Collect File items, the where-clause must include at least one condition on Collect File items and one condition on database items.
- If the report definition does not include any Collect File items, the where-clause can include conditions on Collect File items in addition to any condition on database items.
- The SAME operator is not allowed.

When you include a where-clause, the REPORT processor scans the Index once to collect potential records for all specified reports. Next, it determines the lowest schema record common to all reports. The processor then builds report records for each report, creating one report record for each occurrence of the lowest common schema record.

For example, consider the following REPORT session:

```
REPORT:
FOR REPORT ABC, PRINT /Report ABC/:
    FOR RECORD, PR C1:
    END REPORT:

FOR REPORT DEF, PRINT /Report DEF/:
    FOR RECORD, PR C301:
    END REPORT:

GENERATE ALL:
```

Since no where-clause is specified, there are more report records generated for report DEF (one for each occurrence of C301 values) than for report ABC (one for each occurrence of C1 values). However, if the GENERATE command had included a where-clause, there would be as many report records generated for report ABC as for report DEF: one for each occurrence of C300, the lowest common schema record.

The example illustrates how you may end up with more records than expected when the reports specified in a GENERATE command do not share the same lowest schema record.

To a large extent, the process involved is similar to a QUEST language PRINT command: the where-clause determines the qualified records, the action-clause selects the records. In the case of a REPORT session, all items specified in all reports form the action-clause. See Appendix B for details.

The REPORT processor offers the advantage that you can include a SELECT IF command within each report definition to narrow the selection process. For example, consider the following REPORT session:

```
REPORT:
FOR REPORT HOURLY,
    SELECT IF C112 EQ HOURLY*:
    PRINT /Report Hourly/:
    FOR RECORD, PR C1, C112:
    END REPORT:

FOR REPORT MONTHLY,
    SELECT IF C112 EQ MONTHLY*:
    PRINT /Report Monthly/:
    FOR RECORD, PR C1, C112:
    END REPORT:

GENERATE ALL WH C112 EXISTS:
```

In this example, the GENERATE command qualifies the potential report records and the SELECT IF commands within each report further select from among those records.

Appendix A

User Exits in the REPORT Processor

You can code external routines and call them from a report definition. User-exit routines may be useful for special calculations, for special testing, for synonym tables, or for table look-up applications.

To pass data between the REPORT processor and a routine, you declare external variables within the report definition. The DECLARE EXTERNAL command (see **DECLARE EXTERNAL Command** on page A-2) allows you to name the external variable, to identify its associated routine, and, optionally, to pass parameters.

If a routine requires that several different item values be passed to it, you must declare a unique external variable for each item. A CALL clause (see **CALL Clause** on page A-4) is then required to pass these separate values to the routine before its final result can be requested.

Whenever an external variable is referenced in the report definition, the associated routine is called. The routine acts on these data and returns a value to the REPORT processor. The returned value is used as the variable value for any further computation or printing.

The routine specified in a DECLARE EXTERNAL command must reside in memory as a compiled, executable program within the SYSTEM 2000 code.

The rest of this appendix covers the following topics:

- declaring and calling external variables within a report definition
- writing and link-editing a routine
- examples.

DECLARING AND CALLING EXTERNAL VARIABLES

As mentioned before, you use the DECLARE EXTERNAL command to declare an external variable and a CALL clause to pass values to a routine.

DECLARE EXTERNAL Command

The DECLARE EXTERNAL command names a variable to be used for passing data to and from a routine.

Format

```
DECLARE [type] EXTERNAL variable = routine [WITH n] [USING unit] :
```

type

```
|CHAR
|DATE
|DECIMAL        (default)
|DOUBLE
|INTEGER
|MONEY
|REAL
|TEXT
```

The type identifies the type of data returned to the REPORT processor by the routine. When the type specification is omitted, the default is DECIMAL. The designation need not be the exact data type returned, only the category. For example, INTEGER, DECIMAL, or MONEY can be used to designate that numeric data will be returned; either CHAR or TEXT can be used for nonnumeric data. If the variable is to be treated as type DATE, you must specify it as such.

variable is the name of the variable. It can be up to six alphanumeric characters and must begin with a letter. The general considerations that are given for component names when a component is declared apply to derived names as well. See *SYSTEM 2000 DEFINE Language, Version 12, First Edition* for details.

Only already-declared variables can be used to define other variables.

unit

```
|item
|cf-item
|variable
|(arithmetic expression)
|*function*
|*function(parameters)
```

Abbreviations:

DECLARE	= DE
DECIMAL	= DEC
EXTERNAL	= EXT
INTEGER	= INT

item	is a schema item name or component number.
cf-item	is a Collect File item name. It must be followed by (CF) if it has the same name as a schema component.
arithmetic expression	is an expression composed of numeric type items or Collect File items, consists of constants, or previously declared variables, and the operators +, -, *, and /. Date constants must be in MM.DD.YY format or as specified by the DATE FORMAT command. Processing begins with the innermost parentheses and works outward. Then * and / are performed, and finally + and -.
function	is a stored function name or component number.
routine	is the name of the routine.
n	is a nonnegative integer passed to the user exit when the corresponding variable is referenced in the report.

The WITH option supplies positive integer parameters to the routine for use as a switch to vary program entry points into the procedural portion of the program.

The USING option provides alternative types of data entry for a single program. The option can refer to item values, values of previously declared variables, and so on.

The output format of any variable is always determined by the type of edit picture specified in the PRINT clause (**Print Clauses** on page 8-9).

Considerations

- External variable names can be identical to database component numbers or names. For example, the name C126 is valid. However, if C126 appears elsewhere in the report definition, it is treated as an external variable name and not as a component number. For instance, `FOR C126` would cause an error.
- A stored function in the USING option must not include any QUEST or REPORT processor system functions in its definition.
- An external variable name must not be used in the SELECT RECORD IF command if more than one value must be passed to it in order to produce a result.
- An external variable name must not be used in the ORDER BY command if it invokes a subroutine that requires more than one value passed to it to produce a result.
- A variable to be printed in a report becomes an undefined value when it falls outside the range: 5.4×10^{-77} and 7.2×10^{73}.

Examples Assume that user-exit NAMECHG converts first names to initial caps or to initials.

```
DECLARE CHAR EXTERNAL INICAP = NAMECHG WITH 1 USING C3:
DECLARE CHAR EXTERNAL INITL  = NAMECHG WITH 2 USING C3:
```

For example,

```
PR L(2,X(20))FORENAME, L(25,X(20)INICAP, L(50,XX)INITL:
```

The output from that command would look as shown below:

```
DAVID                 David                 D.
LEOPOLD               Leopold               L.
DARRYL                Darryl                D.
FREDDIE               Freddie               F.
```

CALL Clause

A CALL clause passes data to a subroutine without receiving any value back. You must use a CALL clause whenever a subroutine requires more than one value to be passed to it in order to arrive at a result.

Format

CALL *variables*

variable is an EXTERNAL variable defined by a DECLARE command.

Your subroutine must store the information passed by the CALL command in its COMMON block. (See **MOVCAR Subroutine** on page A-9)

Examples Suppose the subroutine UCODE1 requires the values of C122, C123, and C124 in the EMPLOYEE database in order to return a final value to the REPORT processor for printing. The DECLARE commands might be

```
DECLARE DECIMAL EXTERNAL AAA = UCODE1 WITH 1 USING C122:
DECLARE DECIMAL EXTERNAL BBB = UCODE1 WITH 2 USING C123:
DECLARE DECIMAL EXTERNAL CCC = UCODE1 WITH 3 USING C124:
```

In order to print the correct value, call UCODE1 three times, each time passing it the value of the USING unit and the third time receiving back the print value. You can accomplish these by issuing

```
CALL AAA,BBB:  PRINT R(11,9(5).99)CCC:
```

As a more complex example, assume UCODE2 is a subroutine requiring both the current C122 value and the average C122 value to arrive at a result. The necessary DECLARE commands could be:

```
DE DEC SUM1 = SUM OF C122:
DE SUM2 = CNT C122:
DE AVG1 = (SUM1/SUM2):
DE DEC EXT VAL1 = UCODE2 WITH 1 USING AVG1:
DE DEC EXT VAL2 = UCODE2 WITH 2 USING C122:
```

In order to print the result returned by UCODE2, you must pass both AVG1 and C122 to the subroutine. Since AVG1 is dependent on system functions SUM and CNT, its correct value is passed only if SUM1 and SUM2 are evaluated by COMPUTE. A correct sequence would be

```
COMPUTE SUM1,SUM2:  CALL VAL1:  PR R(11,9(5).99) VAL2:
```

WRITING AND LINK-EDITING A ROUTINE

REPORT processor user-exit routines must be written in the FORTRAN, COBOL, PL/I, or Assembler language. The name of the routine must be one to six characters for a FORTRAN routine, one to seven characters for a PL/I routine, and one to eight characters for a COBOL or Assembler routine.

Regardless of the language you write the routine in, the program must include two structures to receive information that is passed between the routine and the REPORT processor.

- The first structure is a status control vector that contains information about the data being passed. The vector consists of six variables described in **Data Passed to the Routine** on page A-6.
- The second structure is a 250-character array in which the value in the USING option (of the DECLARE EXTERNAL command) is passed to the routine, and in which the routine places the value to be returned to the REPORT processor.

In COBOL, these structures are defined in the LINKAGE SECTION, as shown below.

```
LINKAGE SECTION.
01 vector array.
    level variable-1 description.
    level variable-2 description.
    level variable-3 description.
    level variable-4 description.
    level variable-5 description.
    level variable-6 description.
01 value array.
    variable  description.
```

In FORTRAN, these structures are defined with the INTEGER and EQUIVALENCE statements, as shown below.

```
SUBROUTINE routine (vector,value)
INTEGER variable  description
EQUIVALENCE variable  description
```

Data Passed to the Routine

The following information is passed to the routine in the vector array:

Variable-1	contains the internal component number (that is, its position in DESCRIBE order) of the component in the USING option of the DECLARE EXTERNAL command. If the USING option does not refer to a single component, this variable is undefined.
Variable-2	contains the component number of the component in the USING option. If the option does not refer to a single component, this variable is undefined.
Variable-3	contains the parameter value specified in the WITH option of the DECLARE EXTERNAL command. If no parameter is specified, the variable is set to -1.
Variable-4	contains an integer to designate the type of data passed to the routine from the REPORT processor:

-1	no value to pass. There is no USING option specified.
0	the USING option refers to an item that may be null or may not occur.
1	for CHAR values.
2	for TEXT values.
3	for DATE values.
4	for INTEGER values.
5	for DECIMAL values.
6	for MONEY values.
8	for REAL values.
9	for DOUBLE values.

Variable-5 and Variable-6	contain the name of the report left-justified, zero filled.

Data Received from the Routine

The following data are passed from the routine to the REPORT processor:

Vector Array Variable-4	Value Array
-1	undefined
0	null or non-occurrence
1,2	data string beginning in byte 0, word 1 and extending to the first binary-zero byte
3	a double-precision floating point number in words 6 and 7 that specifies the date in Gregorian form
4	INTEGER value in double-precision floating-point format in words 6 and 7
5,6	DECIMAL and MONEY value in double-precision floating-point format in words 6 and 7
8	REAL value in double-precision floating-point format in words 6 and 7
9	DOUBLE values in double-precision floating-point format in words 6 and 7.

You can modify both the value array and Variable-4 in the vector array only if the value array is consistent with Variable-4, and if Variable-4 is not being changed to a category different from the type in the DECLARE EXTERNAL command. For example, changing from numeric to non-numeric or from DATE to some other type is not allowed.

To illustrate these two structures, consider the example shown under **DECLARE EXTERNAL Command** on page A-2:

```
REPORT:
FOR REPORT SAMPLE:
    .
    .
    .
DECLARE CHAR EXTERNAL INICAP = NAMECHG WITH 1 USING C2:
    .
    .
    .
PR L(2,X(20))LAST NAME, L(25,X(20)INICAP, L(50,XX)INITL:
    .
    .
    .
```

The PRINT clause passes the following information in the vector array:

```
variable-1=2
variable-2=2
variable-3=1
variable-4=1
variable-5/6=SAMPLE
```

Considerations

- A routine can reference other routines or entry points, but these must not be referenced in other DECLARE EXTERNAL commands; they must be included as part of some link-edited load module identified by the same name as the main routine.
- The routine must not do I/O operations on any of the SYSTEM 2000 user files, that is, the Data File, Command File, Message File, or Report File.
- You are responsible for closing all your files.
- You cannot have more than 20 unique user-exit modules for each REPORT/GENERATE session.
- Any printing performed by your routine does not increment the LINE and PAGE counters maintained by the REPORT processor. Therefore, you must use SKIP clauses for the page breaks to occur at page boundaries.

Procedures for Link-Editing REPORT User Exits

The routine must exist as a link-edited load module in the JOBLIB or STEPLIB data set(s) or in the operating system's link library. The load module so identified is brought into main storage dynamically by SYSTEM 2000 software at command parsing time via the "LOAD SVC." The entry point name in the load module must be a member name or an alias in a directory of a partitioned data set. If the specified entry point cannot be located, an error message is issued.

The module is "loaded" only once for each REPORT/GENERATE session. Such a module is in its initial state the first time it is called within a run unit. On all other entries into the called program, the state of the called program remains unchanged from its state when last executed.

SAMPLE COBOL ROUTINE

Here is a sample COBOL REPORT user-exit routine and the SYSTEM 2000 commands utilizing the routine.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  SAMPLE.
    .
    .
    .
ENVIRONMENT DIVISION.
    .
    .
    .
DATA DIVISION.
WORKING-STORAGE SECTION.
01  FP5  PIC S9(18) COMP-2 SYNC VALUE +5.0E+0.
LINKAGE SECTION.
01  VECTOR.
    02 V1 PIC S9(9) COMP  SYNC.
    02 V2 PIC S9(9) COMP  SYNC.
    02 V3 PIC S9(9) COMP  SYNC.
```

```
        02 V4 PIC S9(9) COMP  SYNC.
        02 V5 PIC X(8).
    01  VAL.
        02 VA PIC X(3).
        02 VB PIC X(1).
        02 VC PIC X(16).
        02 VD PIC X(30).
    01  VALFP REDEFINES VAL.
        02 FILLER PIC X(20).
        02 VFP    PIC S9(18)   COMP-2.

    PROCEDURE DIVISION USING VECTOR VAL.
    BEGIN-EXECUTION.
        IF V4  <  3 GO TO CHAR-SUB.
        ADD FP5 TO VFP.                      (adds 5 to numeric in "VAL")
        GO TO RETURN-POINT

    CHAR-SUB.
        MOVE '-' TO VB.                      (replaces one character)
    RETURN-POINT.
        GOBACK.
```

A sample report using the above routine is shown next.

```
REPORT:
FOR REPORT TEST,
  PHYSICAL PAGE IS 50 BY 10:
  DECLARE CHAR EXTERNAL ROOM=SAMPLE USING C10:
  DECLARE INTEGER EXTERNAL SEC=SAMPLE USING C13:

FOR C10,
  PRINT /C10=/, L(6,X(15)) C10, (23)/ROOM=/, L(30,X(15)) ROOM:
  PRINT /C13=/, R(6,9(3)) C13, (23)/SEC=/, R(30,9(3))SEC:
END:
GENERATE TEST WHERE C10 EQ 410 XT369:
```

The resulting output is

```
C10= 410 XT369          ROOM=  410-XT369
C13= 224                SEC=229
```

MOVCAR SUBROUTINE

The MOVCAR subroutine is available to handle strings.

CALL MOVCAR (CTRL,*from-array*,*to-array*)

CTRL is a 5-word array defined as follows:

CTRL(1) is the maximum number of characters to be moved.

CTRL(2)	is the mode of move 0 move exactly CTRL(1) characters 1 stop at first zero byte
CTRL(3)	is the starting character position in *from-array* (beginning with 0)
CTRL(4)	is the starting character position in *to-array* (beginning with 0)
CTRL(5)	is set to 1.

Appendix B

REPORT Processor Execution

This appendix describes the three steps used by the REPORT processor to produce a report.

STEP 1: PARSE THE INDIVIDUAL REPORTS

The REPORT processor first parses the individual reports and converts them to internal form.

The REPORT processor resolves name conflicts among item, variable, and Collect File item (CF-item) names. The order of precedence is variable, database item, and CF-item. If you need to give the CF-item a higher precedence, you must append (CF) to the CF-item name.

At the conclusion of Step 1, each report has associated with it two lists, one containing the set of all unique database components used in the report and the other the set of all unique Collect File item names used in the report.

STEP 2: PROCESS THE GENERATE COMMAND

In the second step, the REPORT processor parses the GENERATE command, determines the lowest common schema record, either processes the where-clause (if any) or makes a full pass of the database, and builds all the report records.

The GENERATE command can have

- no where-clause
- a where-clause that references only database components
- a where-clause that references both database components and CF items.

The procedure used to build the report records for the entire set of reports (which have already been converted to internal form in Step 1) depends upon which of these three choices is specified in the GENERATE command. The procedure is different for each case, and each procedure is explained below in detail.

Each case must take into consideration the choice of items and components used by each report. A report can have

- neither CF items nor database components
- only database components
- only CF items
- both CF items and database components.

No Where-Clause

With no where-clause, each report must use either all CF items or all database components.

The REPORT processor determines whether any report to be executed contains references to both CF items and database components. If so, it issues an error message and omits the report from the set of reports to be produced.

The reports are then divided into two groups: one group contains reports that use CF items, the other group contains reports that use database components. Each group is processed individually (as explained below), with the final set of report records being those created by each group.

To build reports with CF items only, the REPORT processor

1. reads the next (or first) record from the Collect File. If there are no more records, then processing for the group ends and processing for the second group starts.

2. builds a report record for the next (or first) report in the group. If there are no more reports in the group, it goes back to 1.

3. goes back to 2.

To construct reports with database components only, the REPORT processor first determines the lowest common schema record in the set of reports. Then the processor

1. locates the next (or first) instance of the lowest common schema record in the database. If there are no more instances, it terminates.

2. builds report records for the next (or first) report. If there are no more reports in the group, it goes back to 1.

3. goes back to 2.

Where-Clause Refers to Database Components Only

The REPORT processor first determines whether any of the reports to be executed contains references to CF items. If so, it issues an error message and deletes the report from the set. It then determines the lowest common schema record of the reports to be executed and constructs the report records as follows:

1. processes the where-clause; that is, using the selection criteria given in the where-clause, it makes a list of the qualified records.

2. normalizes the list of qualified data records to the lowest common schema record, if necessary; that is, it produces the list of selected records.

3. builds report records for the next (or first) report, using the list of selected records as input. If there are no more reports, it terminates.

4. goes back to 3.

Where-Clause Refers to Database Components and CF Items

To construct the report records, the REPORT processor first determines the lowest common schema record of the reports to be executed. Then the processor

1. executes the Collect File where-clause. The result is a set of lists of selected records, one list for each Collect File record. (A detailed description of how the where-clause is executed with the Collect File is given in *SYSTEM 2000 QUEST Language and System-Wide Commands, Version 12, First Edition*)

2. reads the next (or first) record from the Collect File. If there are no more records, it terminates.

3. reads the next (or first) list of selected records and normalize inclusively.

4. builds report records for the next (or first) report. If there are no more reports, then goes back to 2.

5. goes back to 4.

STEP 3: PRODUCE THE REPORTS

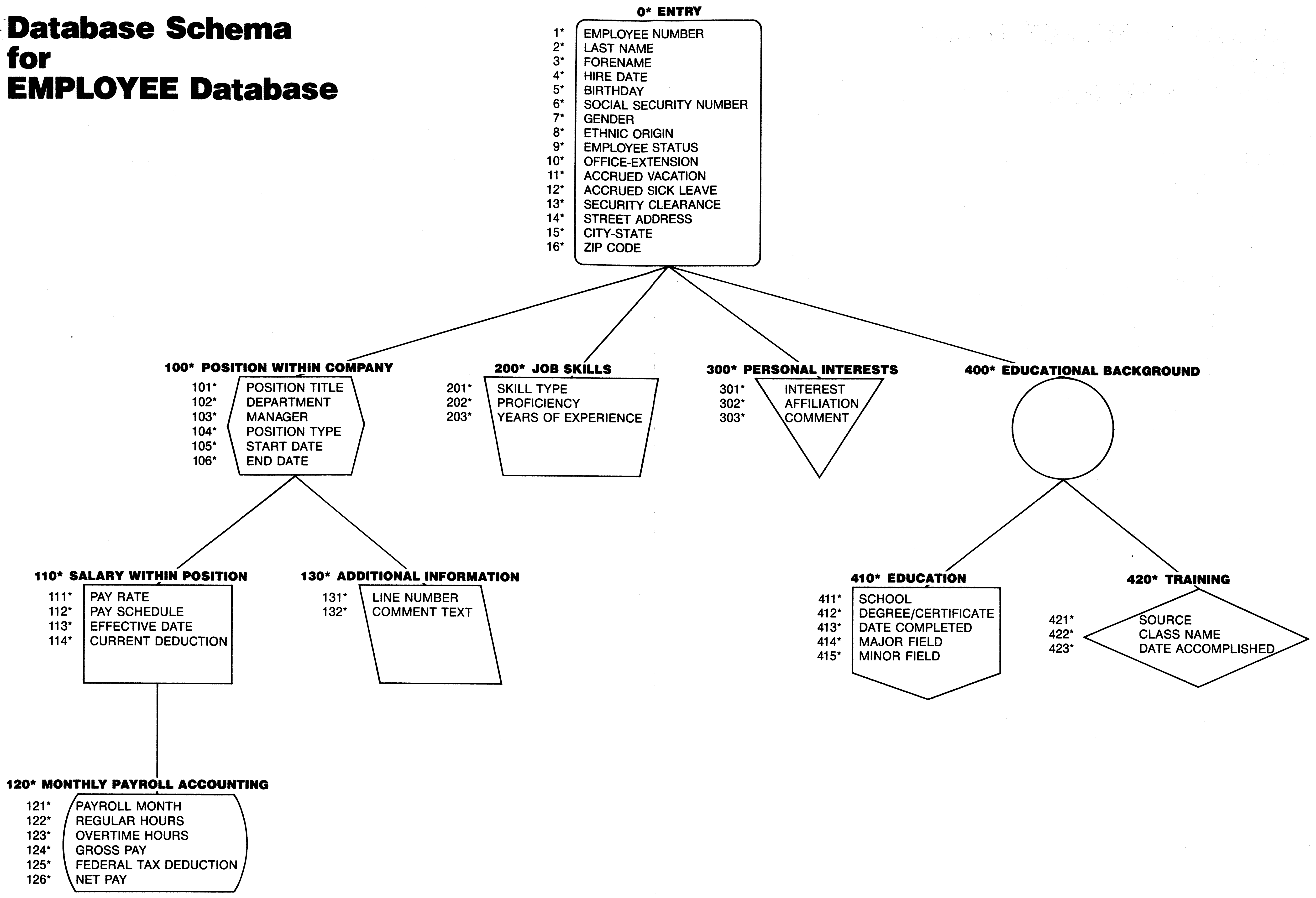
The third step sorts the report records as necessary. Then the output for each report is produced.

Index

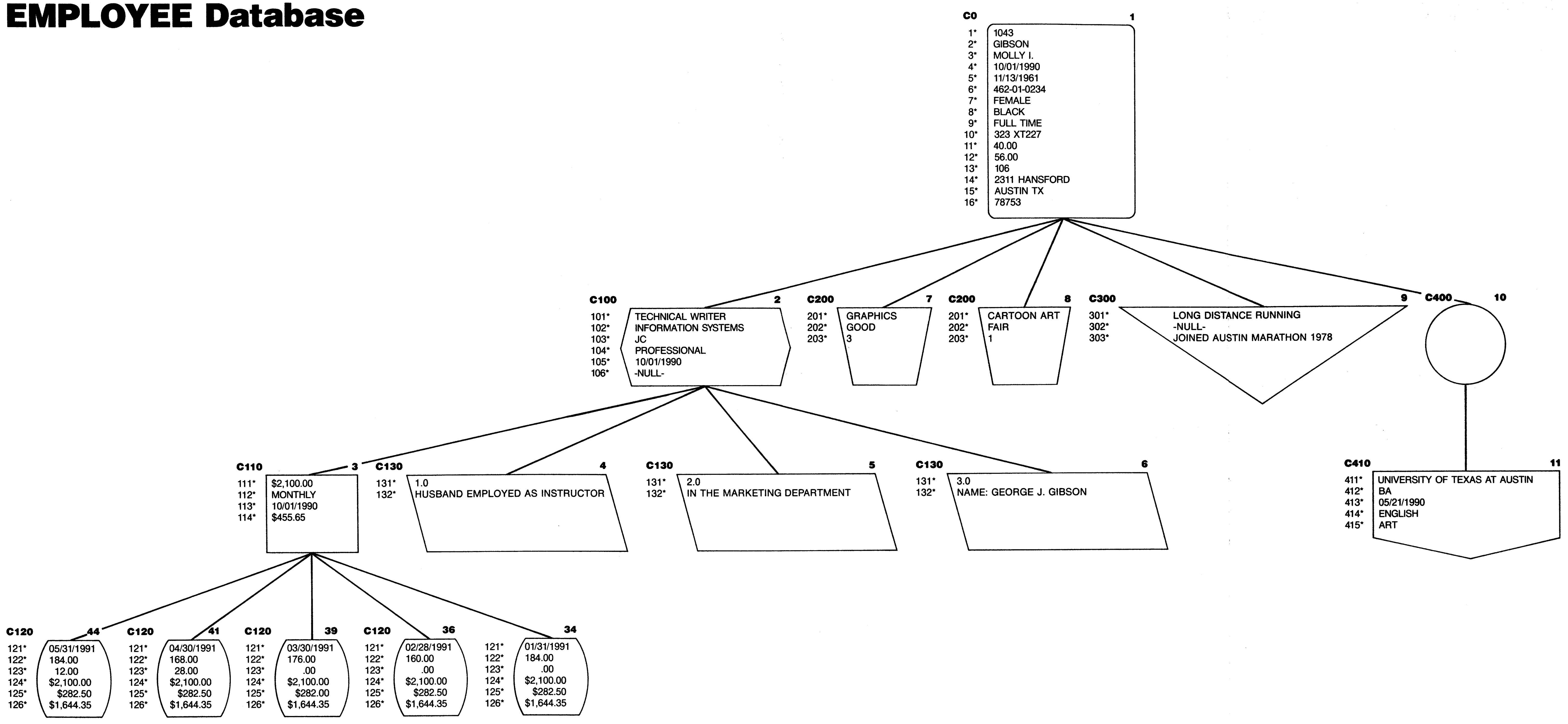
Actual DESCRIBE Output from EMPLOYEE Database

```
DESCRIBE:
SYSTEM RELEASE NUMBER XXX
DATABASE NAME IS         EMPLOYEE
DEFINITION NUMBER                  3
DATA BASE CYCLE NUMBER             1
       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*   GENDER (CHAR X(6) WITH MANY FUTURE OCCURRENCES)
       8*   ETHNIC ORIGIN (CHAR X(9) WITH MANY 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 OCCURRENCES)
      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 OCCURRENCES)
       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 FUTRUE 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)
         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)
```

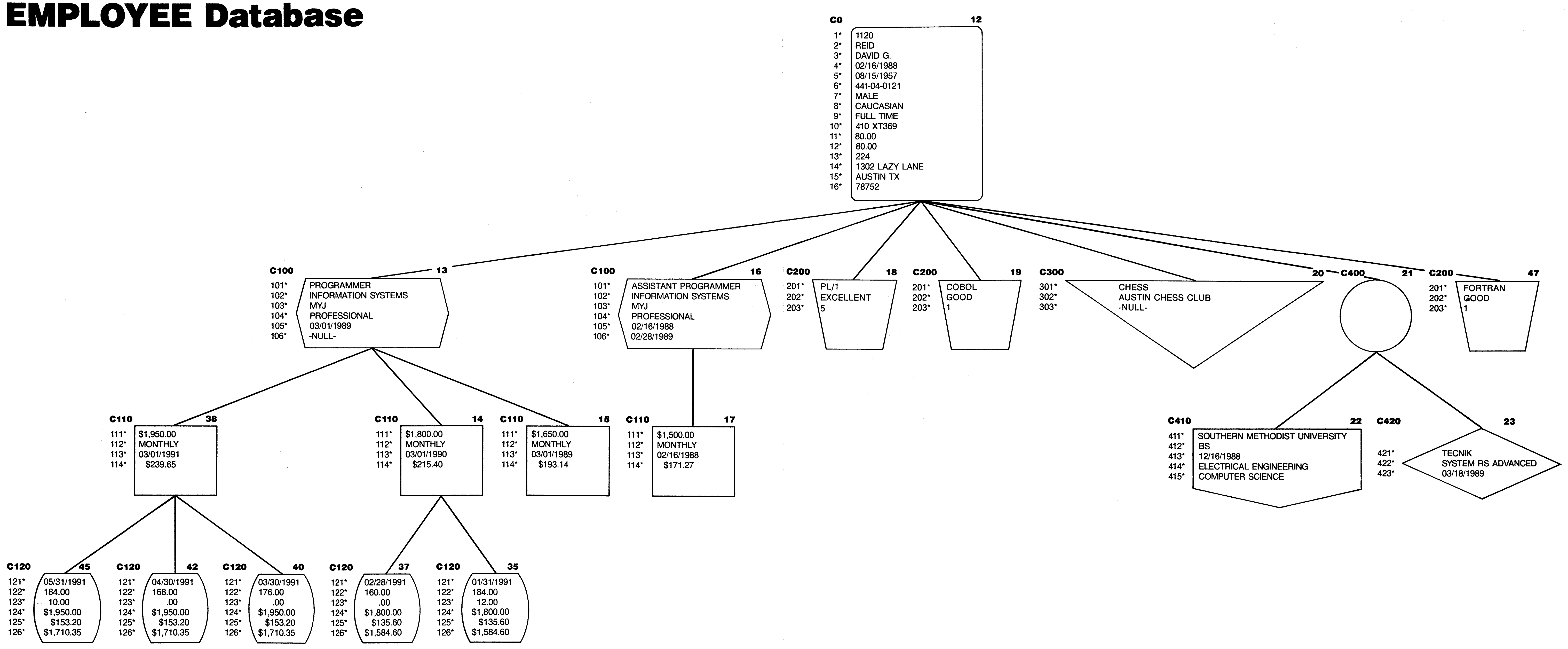
Database Schema for EMPLOYEE Database



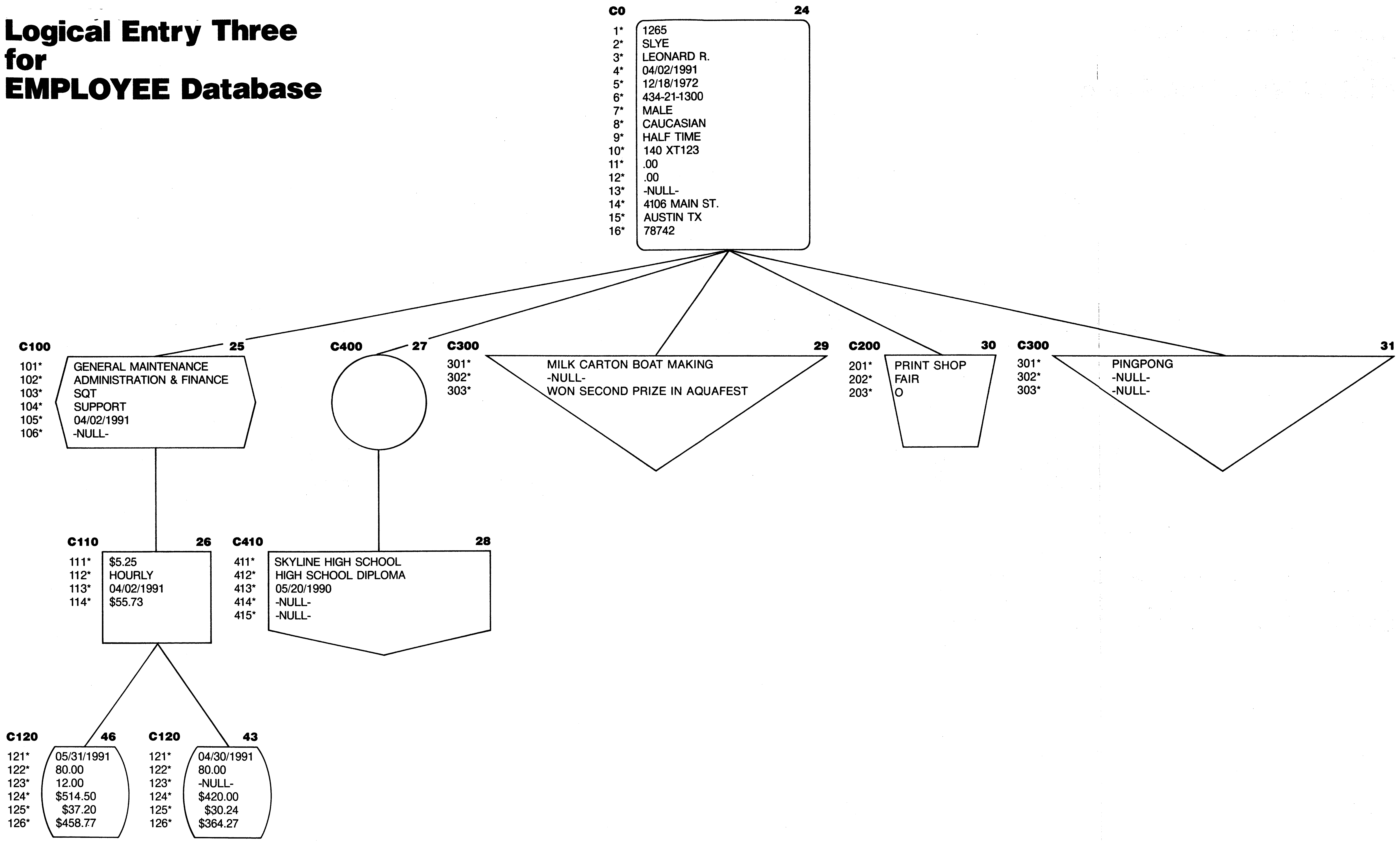
Logical Entry One for EMPLOYEE Database



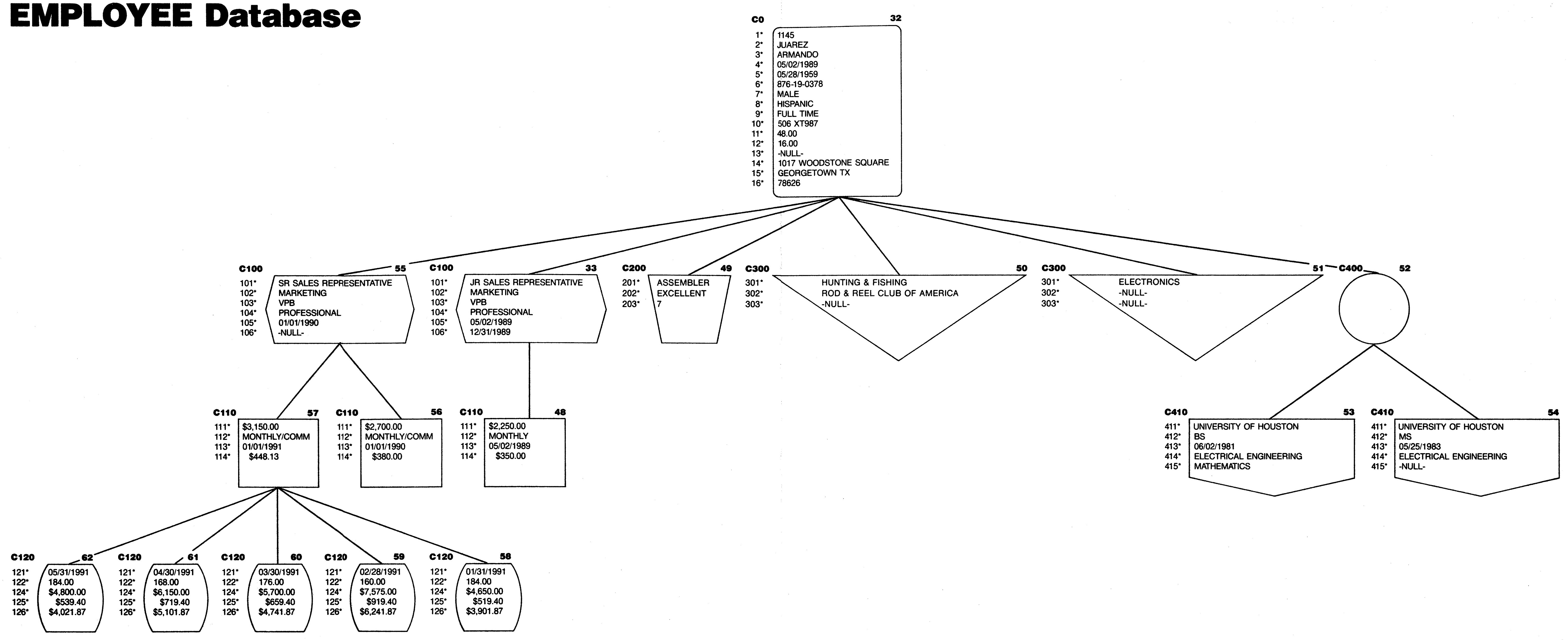
Logical Entry Two for EMPLOYEE Database



Logical Entry Three for EMPLOYEE Database



Logical Entry Four for EMPLOYEE Database



Your Turn

If you have comments or suggestions about SYSTEM 2000 software or *SYSTEM 2000® REPORT Language, Version 12, First Edition*, please send them to us on a photocopy of this page.

Please return the photocopy to the Publications Division (for comments about this book) or the Technical Support Division (for suggestions about the software) at SAS Institute Inc., P.O. Box 200075, Austin, TX 78720-0075.