

ATTN: CHARLIE GIBBS

01197
CAV208M45541 UP 9209

UAS

SPERRY UNIVAC
1 - 1818 CORNWALL STREET
VANCOUVER B C V6J 1C7 CANADA

CAV

**PUBLICATIONS
REVISION**

Operating System/3 (OS/3)

Information Management
System (IMS)

Data Definition and UNIQUE

User Guide

UP-9209

This Library Memo announces the release and availability of "SPERRY UNIVAC® Operating System/3 (OS/3) Information Management System (IMS) Data Definition and UNIQUE User Guide", UP-9209.

The Information Management System (IMS) Data Definition and UNIQUE User Guide, UP-9209, is one of five books replacing the IMS 90 Applications User Guide/Programmer Reference, UP-8614, Rev. 1. Other manuals replacing UP-8614 are:

- IMS Concepts and Facilities, UP-9205
- IMS Action Programming in RPG II User Guide, UP-9206
- IMS Action Programming in COBOL and Basic Assembly Language (BAL) User Guide, UP-9207
- IMS Terminal Users Guide, UP-9208

This manual explains data definitions and describes how to use UNIQUE. It is presented in four parts as follows:

1. OVERVIEW

Section 1. Introduction

2. DATA DEFINITIONS

Section 2. Defined File Structure

Section 3. Writing Data Definitions

Section 4. Data Definition Examples

Section 5. The Data Definition Processor

3. UNIQUE

Section 6. Introduction to UNIQUE

Section 7. UNIQUE Commands

| LIBRARY MEMO ONLY | LIBRARY MEMO AND ATTACHMENTS | THIS SHEET IS |
|--------------------------------|---|---|
| Mailing Lists BZ, CZ and MZ | Mailing Lists A00, A07, A08, B00, B07, B08, 18, 18U, 19, 19U, 20, 20U, 21, 21U, 28U, 29U, 75, 75U, 76, and 76U (Cover and 216 pages) | Library Memo for UP-9209 RELEASE DATE: September, 1982 |

4. APPENDIXES

Appendix A. Format Presentation and Coding Rules

Appendix B. Reserved Words

Appendix C. Data Definition Processor Diagnostics

Appendix D. UNIQUE Lexicon

Appendix E. Data Definitions for UNIQUE Examples

The complete titles and ordering numbers of the manuals that form the IMS library are:

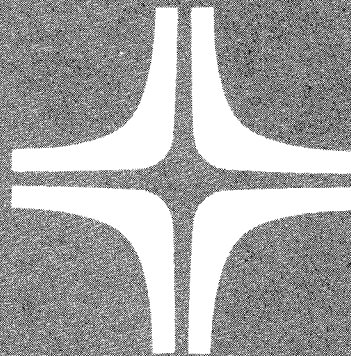
- Information Management System (IMS) System Support Functions User Guide, UP-8364, Rev. 7
- Information Management System (IMS) Concepts and Facilities, UP-9205
- Information Management System (IMS) Action Programming in RPG II User Guide, UP-9206
- Information Management System (IMS) Action Programming in COBOL and Basic Assembly Language (BAL) User Guide, UP-9207
- Information Management System (IMS) Terminal Users Guide, UP-9208
- Information Management System (IMS) Data Definition and UNIQUE User Guide, UP-9209
- IMS/DMS Interface User Guide, UP-8748, Rev. 1

Additional copies may be ordered by your local Sperry Univac representative.

Information Management System (IMS)

Data Definition and UNIQUE

OS/3



User Guide

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Corporation.

FASTRAND, SPERRY UNIVAC, UNISCOPE, UNISERVO, and UNIVAC are registered trademarks of the Sperry Corporation. ESCORT, MAPPER, PAGEWRITER, PIXIE, and UNIS are additional trademarks of the Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PAGE STATUS SUMMARY

ISSUE: UP-9209
RELEASE LEVEL: 8.0 Forward

| Part/Section | Page Number | Update Level | Part/Section | Page Number | Update Level | Part/Section | Page Number | Update Level |
|--------------------|-------------|--------------|--------------|-------------|--------------|--------------|-------------|--------------|
| Cover/Disclaimer | | | | | | | | |
| PSS | 1 | | | | | | | |
| Acknowledgment | 1 | | | | | | | |
| Preface | 1 thru 4 | | | | | | | |
| Contents | 1 thru 8 | | | | | | | |
| PART 1 | Title Page | | | | | | | |
| 1 | 1 thru 4 | | | | | | | |
| PART 2 | Title Page | | | | | | | |
| 2 | 1 thru 5 | | | | | | | |
| 3 | 1 thru 78 | | | | | | | |
| 4 | 1 thru 15 | | | | | | | |
| 5 | 1 thru 10 | | | | | | | |
| PART 3 | Title Page | | | | | | | |
| 6 | 1 thru 3 | | | | | | | |
| 7 | 1 thru 54 | | | | | | | |
| PART 4 | Title Page | | | | | | | |
| Appendix A | 1 thru 4 | | | | | | | |
| Appendix B | 1 thru 3 | | | | | | | |
| Appendix C | 1, 2 | | | | | | | |
| Appendix D | 1 thru 4 | | | | | | | |
| Appendix E | 1, 2 | | | | | | | |
| Index | 1 thru 12 | | | | | | | |
| User Comment Sheet | | | | | | | | |
| | | | | | | | | |

All the technical changes are denoted by an arrow (→) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (→) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.



Acknowledgment

We are indebted to the many systems analysts and staff members of Sperry Univac branch offices and customer organizations who helped us develop the OS/3 IMS library. They gave us suggestions, answered questions, reviewed the manuals, and provided us with "real-life" programming examples. The customer organizations assisting us include:

- Gay and Taylor Insurance Adjustors, Winston-Salem, NC
- Penn Ventilator Company, Philadelphia, PA
- Victor Valley Community College District, Victorville, CA

The Sperry Univac organizations assisting us include:

- Los Angeles Access Center, Customer Support Services, Los Angeles, CA
- Charlotte Commercial Branch, Raleigh Office, Raleigh, NC
- Charlotte Commercial Branch, Greensboro Office, Greensboro, NC
- Minneapolis Marketing Branch, Minneapolis, MN
- Wellesley General Branch, Wellesley, MA
- Philadelphia Manufacturing Branch, Wayne, PA
- Des Moines Marketing Branch, West Des Moines, IA
- System 80 Benchmark and Demonstration Services, Blue Bell, PA



Preface

This manual is one of a series designed to instruct and guide you in using the SPERRY UNIVAC Information Management System (IMS) for Operating System/3 (OS/3). It describes data definitions for use with the uniform inquiry update element (UNIQUE) or your action programs and explains how to use UNIQUE.

This manual consists of two major topics. First, it tells you how to write a data definition. The data definition language is similar to COBOL, so it is easier to write a data definition when you already know COBOL.

Before you start writing data definitions, you should have a basic understanding of IMS, how it operates, and what you (or the IMS administrator) need to do to make it operational. This information is in the IMS concepts and facilities manual, UP-9205 (current version).

Second, it tells you how to use UNIQUE commands to access your defined files. This part of the manual can also be used as a training guide for terminal operators using UNIQUE.

To access defined files from action programs, you also need the current version of the IMS action programming in RPG II user guide, UP-9206 or the IMS action programming in COBOL and basic assembly language (BAL) user guide, UP-9207.

The information in this manual is presented in four parts:

PART 1. OVERVIEW

- Section 1. Introduction

Describes the purpose of a data definition, the concepts it is based upon, and its relationship to UNIQUE.

PART 2. DATA DEFINITIONS

- Section 2. Defined File Structure

Describes the makeup and types of defined files.

- Section 3. Writing Data Definitions

Describes data definition structure and explains how to use source statements.

- Section 4. Data Definition Examples

Provides extended examples of data definitions and the defined files they produce. Shows how defined files are derived from source files, how they appear to UNIQUE and action programs, and the record areas required in action programs.

- Section 5. The Data Definition Processor

Describes how to execute the data definition processor and the output listings you receive from the processor.

PART 3. UNIQUE

- Section 6. Introduction to UNIQUE

Provides a brief description of UNIQUE commands, passwords, and dialogs.

- Section 7. UNIQUE Commands

Describes the UNIQUE commands and gives extensive examples of their use.

PART 4. APPENDIXES

- Appendix A. Format Presentation and Coding Rules

Describes format and coding rules for data definitions and UNIQUE.

- Appendix B. Reserved Words

Lists data definition reserved words.

- Appendix C. Data Definition Processor Diagnostics

Lists error diagnostics issued by the data definition processor.

- Appendix D. UNIQUE Lexicon

Lists the language elements in the standard UNIQUE lexicon.

- Appendix E. Data Definitions for UNIQUE Examples

Gives data definitions for the defined files used in the examples in Section 7.

As one of a series, this manual is designed to guide you in programming and using the OS/3 information management system. Depending on your need, you should also refer to the current versions of other manuals in the series. Complete manual names, their ordering numbers, and a general description of their contents and use are as follows:

- Information management system (IMS) concepts and facilities, UP-9205

Describes the basic concepts of IMS and the facilities that IMS offers.

- Information management system (IMS) system support functions user guide, UP-8364

Describes the procedures to generate, initiate, and recover an online IMS system.

- Information management system (IMS) action programming in RPG II user guide, UP-9206

Describes how to write action programs in RPG II, with extensive examples.

- Information management system (IMS) action programming in COBOL and basic assembly language (BAL) user guide, UP-9207

Describes how to write action programs in COBOL and BAL, with extensive examples.

- Information management system (IMS) terminal users guide, UP-9208

Describes terminal operating procedures, standard and master terminal commands, and special purpose IMS transaction codes. Also includes UNIQUE command formats with brief descriptions. The manual is in easel format for ease of use at the terminal.

- **IMS/DMS interface user guide, UP-8748**

Describes how to access a data base management system (DMS) data base from IMS.

Contents

PAGE STATUS SUMMARY

ACKNOWLEDGMENT

PREFACE

CONTENTS

PART 1. OVERVIEW

1. INTRODUCTION

| | | |
|------|--|-----|
| 1.1. | CONCEPT AND PURPOSE | 1-1 |
| 1.2. | CREATING DEFINED FILES | 1-2 |
| 1.3. | ACCESSING DEFINED FILES THROUGH UNIQUE | 1-3 |
| 1.4. | DEFINED RECORD MANAGEMENT | 1-3 |

PART 2. DATA DEFINITIONS

2. DEFINED FILE STRUCTURE

| | | |
|------|----------------------------|-----|
| 2.1. | DEFINED FILES | 2-1 |
| 2.2. | DEFINED RECORDS | 2-2 |
| 2.3. | DATA DEFINITION RECORDS | 2-3 |
| 2.4. | STRUCTURE OF DEFINED FILES | 2-3 |

3. WRITING DATA DEFINITIONS

| | | |
|-------|---|------|
| 3.1. | DATA DEFINITION LANGUAGE | 3-1 |
| 3.2. | DATA DEFINITION STRUCTURE | 3-3 |
| 3.3. | IDENTIFICATION DIVISION | 3-4 |
| 3.4. | DATA DIVISION | 3-4 |
| 3.5. | DEFINITION DIVISION | 3-7 |
| 3.6. | DEFINED FILE DEFINITION | 3-8 |
| 3.7. | NAMING THE DEFINED FILE (DEFINED FILE STATEMENT) | 3-10 |
| 3.8. | DEFINED RECORD DEFINITION | 3-12 |
| 3.9. | DESCRIBING A DEFINED RECORD'S PRIMARY PART (DEFINED RECORD STATEMENT) | 3-14 |
| 3.10. | NAMING THE DEFINED RECORD (DEFINED RECORD CLAUSE) | 3-15 |
| 3.11. | IDENTIFYING THE SOURCE OF THE PRIMARY PART (FROM CLAUSE) Specifying One Record as the Source | 3-16 |
| | Specifying a Sequence of Records as the Source | 3-17 |
| | Specifying a Repeating Group as the Source | 3-20 |
| 3.12. | DEFINING THE RECORD TYPE (TYPE CLAUSE) | 3-21 |
| 3.13. | IDENTIFYING THE PARENT OF A CHILD DEFINED RECORD (PARENT CLAUSE) | 3-22 |
| 3.14. | DESIGNATING THE PROCESSING ORDER OF FRATERNAL RECORDS (PREFIX CLAUSE) | 3-25 |
| 3.15. | LOCATING THE SOURCE OF A CHILD DEFINED RECORD (POINTER CLAUSE) | 3-27 |
| 3.16. | LOCATING THE SOURCE OF A DEFINED RECORD (FOLLOWS CLAUSE) | 3-29 |
| 3.17. | ACCESSING RECORDS ACCORDING TO TYPE (FILL KEY CLAUSE) | 3-31 |
| 3.18. | ALLOWING RECORD ADDITIONS AND DELETIONS (ALLOW ADD AND DELETE CLAUSE) | 3-33 |
| 3.19. | ITEM DEFINITION | 3-34 |
| 3.20. | DEFINING AN IDENTIFIER (IDENTIFIER STATEMENT) | 3-37 |
| 3.21. | NAMING THE IDENTIFIER (IDENTIFIER CLAUSE) | 3-38 |
| 3.22. | SPECIFYING A VALUE RANGE FOR THE IDENTIFIER (VALUE CLAUSE) | 3-40 |

| | | |
|-------|--|------|
| 3.23. | DEFINING OTHER ITEMS IN THE DEFINED RECORD (ITEM STATEMENT) | 3-41 |
| 3.24. | NAMING THE ITEM (ITEM CLAUSE) | 3-42 |
| 3.25. | PREVENTING ITEM DISPLAY (HIDDEN OPTION) | 3-43 |
| 3.26. | SPECIFYING A REQUIRED ITEM (MUST ADD OPTION) | 3-44 |
| 3.27. | ALLOWING CHANGES TO THE ITEM (ALLOW CHANGE OPTION) | 3-45 |
| 3.28. | SPECIFYING A VALUE RANGE FOR THE ITEM (VALUE CLAUSE) | 3-46 |
| 3.29. | INCLUDING PREVIOUSLY DEFINED ITEMS IN THE DEFINED RECORD (ALSO CLAUSE) | 3-47 |
| 3.30. | SUPPLEMENT DEFINITION | 3-48 |
| 3.31. | NAMING THE SUPPLEMENT (SUPPLEMENT CLAUSE) | 3-49 |
| 3.32. | IDENTIFYING THE SOURCE OF THE SUPPLEMENT (FROM CLAUSE) | 3-50 |
| | Specifying One Record as the Source | 3-50 |
| | Specifying a Repeating Group as the Source | 3-51 |
| 3.33. | LOCATING THE SOURCE OF THE SUPPLEMENT (POINTER CLAUSE) | 3-53 |
| 3.34. | ACCESSING RECORDS ACCORDING TO TYPE (FILL KEY CLAUSE) | 3-56 |
| 3.35. | SPECIFYING THE EFFECTS OF DEFINED RECORD CHANGES (ROLE IN UPDATE CLAUSE) | 3-59 |
| | Specifying CONTROLLING | 3-60 |
| | Specifying CONTROLLED | 3-61 |
| | Specifying NEUTRAL | 3-61 |
| 3.36. | ITEM DEFINITION | 3-62 |
| 3.37. | NAMING THE ITEM (ITEM CLAUSE) | 3-63 |
| 3.38. | PREVENTING ITEM DISPLAY (HIDDEN OPTION) | 3-63 |
| 3.39. | SPECIFYING A REQUIRED ITEM (MUST ADD OPTION) | 3-64 |
| 3.40. | ALLOWING CHANGES TO THE ITEM (ALLOW CHANGE OPTION) | 3-65 |
| 3.41. | SPECIFYING A VALUE RANGE FOR THE ITEM (VALUE CLAUSE) | 3-65 |
| 3.42. | INCLUDING PREVIOUSLY DEFINED ITEMS IN THE DEFINED RECORD (ALSO CLAUSE) | 3-66 |
| 3.43. | SUBRECORD DEFINITION | 3-67 |
| 3.44. | NAMING THE SUBRECORD (SUBRECORD CLAUSE) | 3-68 |
| 3.45. | INCLUDING PREVIOUSLY DEFINED ITEMS IN THE SUBRECORD (OF CLAUSE) | 3-69 |

| | | |
|-------|---|------|
| 3.46. | ALLOWING SUBRECORD ADDITIONS AND DELETIONS (ALLOW ADD AND DELETE CLAUSE) | 3-69 |
| 3.47. | SUBITEM DEFINITION | 3-70 |
| 3.48. | NAMING THE IDENTIFIER (IDENTIFIER CLAUSE) | 3-71 |
| 3.49. | NAMING THE ITEM (ITEM CLAUSE) | 3-72 |
| 3.50. | SPECIFYING A REQUIRED ITEM (MUST ADD OPTION) | 3-73 |
| 3.51. | ALLOWING CHANGES TO THE ITEM (ALLOW CHANGE OPTION) | 3-74 |
| 3.52. | SPECIFYING A VALUE RANGE FOR THE ITEM (VALUE CLAUSE) | 3-75 |
| 3.53. | SUBFILE DEFINITION | 3-76 |
| 3.54. | NAMING THE SUBFILE (SUBFILE STATEMENT) | 3-76 |
| 3.55. | IDENTIFYING RECORDS INCLUDED IN THE SUBFILE (CONTAINS CLAUSE) | 3-78 |
| | | |
| 4. | DATA DEFINITION EXAMPLES | |
| | | |
| 4.1. | EXAMPLE OF A SIMPLE DEFINED FILE | 4-2 |
| 4.2. | EXAMPLES OF HIERARCHICAL RECORDS IN DEFINED FILES | 4-4 |
| | Parent-Child Defined Records Using Several Record Types as a Source | 4-4 |
| | Parent-Child Defined Records Using a Repeating Group Item as a Source | 4-6 |
| | Parent-Child Defined Records Using Two Indexed Files as Sources | 4-7 |
| | Defined File Resulting from Different File Sources | 4-8 |
| 4.3. | EXAMPLE OF SUPPLEMENTS IN A DEFINED FILE | 4-10 |
| 4.4. | EXAMPLE OF A SUBFILE | 4-12 |
| 4.5. | EXAMPLE OF INTERRELATED DEFINED FILES | 4-14 |
| | | |
| 5. | THE DATA DEFINITION PROCESSOR | |
| | | |
| 5.1. | EXECUTING THE DATA DEFINITION PROCESSOR | 5-1 |
| 5.2. | DATA DEFINITION PROCESSOR OPTIONS | 5-2 |
| | List Options | 5-3 |
| | Source and Copy Library Input Options | 5-4 |
| 5.3. | EXECUTION RUN STREAMS | 5-4 |
| 5.4. | DATA DEFINITION OUTPUT LISTING | 5-7 |
| 5.5. | ERROR PROCESSING BY THE DATA DEFINITION PROCESSOR | 5-8 |

PART 3. UNIQUE

6. INTRODUCTION TO UNIQUE

| | | |
|------|----------------------------|-----|
| 6.1. | SUMMARY OF UNIQUE COMMANDS | 6-1 |
| 6.2. | PASSWORDS AND UNIQUE | 6-3 |
| 6.3. | UNIQUE DIALOG | 6-3 |

7. UNIQUE COMMANDS

| | | | |
|-------|---|-----------|------|
| 7.1. | UNIQUE FORMATS AND RULES FOR ENTERING COMMANDS | | 7-1 |
| 7.2. | DATA USED IN OUR EXAMPLES | | 7-2 |
| | INVFILE File | | 7-2 |
| | SALES File | | 7-3 |
| | Identifiers and Item Names | | 7-4 |
| 7.3. | OPENING A UNIQUE DIALOG | (OPEN) | 7-5 |
| 7.4. | ENDING THE UNIQUE TRANSACTION | (CLOSE) | 7-5 |
| 7.5. | DISPLAYING A RECORD | (DISPLAY) | 7-7 |
| | Displaying a Parent Record | | 7-7 |
| | Displaying a Child Record | | 7-7 |
| | Displaying More than One Record | | 7-8 |
| | Replacing Identifiers with Hyphens | | 7-9 |
| 7.6. | SELECTING THE NEXT RECORD | (NEXT) | 7-9 |
| 7.7. | DELETING A RECORD | (DELETE) | 7-11 |
| | Deleting One Record | | 7-11 |
| | Canceling a Deletion | | 7-12 |
| | Deleting More than One Record | | 7-12 |
| 7.8. | AUTHORIZING AN UPDATE FUNCTION | (OK) | 7-13 |
| 7.9. | CANCELING AN UPDATE FUNCTION | (CANCEL) | 7-13 |
| 7.10. | ADDING A RECORD | (ADD) | 7-15 |
| 7.11. | DISPLAY FORMAT OF THE ADD COMMAND | | 7-15 |
| 7.12. | HARD-COPY FORMAT OF THE ADD COMMAND | | 7-20 |
| 7.13. | CHANGING A RECORD | (CHANGE) | 7-24 |
| 7.14. | DISPLAY FORMAT OF THE CHANGE COMMAND | | 7-24 |
| 7.15. | HARD-COPY FORMAT OF THE CHANGE COMMAND | | 7-27 |
| 7.16. | LISTING THE RECORDS IN A FILE | (LIST) | 7-30 |

| | | | |
|-------|---|----------|------|
| 7.17. | LISTING COMPLETE CONTENTS OF ALL RECORDS (UNQUALIFIED LIST) | | 7-30 |
| 7.18. | SELECTING ITEMS OR RECORDS FOR LISTING (DISPLAY-CONTENT-SPECIFICATION) | | 7-31 |
| | Selecting Record Items for Listing (Item-names) | | 7-32 |
| | Selecting Complete Records for Listing (Record-name, ALL) | | 7-33 |
| | Selecting Subrecords for Listing (Subrecord-name) | | 7-36 |
| 7.19. | SELECTING RECORDS THAT MEET A CONDITION (IF CLAUSE) | | 7-37 |
| 7.20. | LISTING A SUBSET OF A HIERARCHICAL FILE (FOR CLAUSE) | | 7-40 |
| 7.21. | SPECIFYING A STARTING POINT FOR SELECTING RECORDS (AFTER/FROM CLAUSE) | | 7-43 |
| 7.22. | GENERATING STATISTICS (STATISTICAL-FUNCTION) | | 7-45 |
| 7.23. | REQUESTING THE NEXT LIST OR DETAIL SCREEN | (MORE) | 7-48 |
| 7.24. | OBTAINING A SECONDARY LISTING | (DETAIL) | 7-49 |
| 7.25. | DISPLAYING RECORD FORMATS AND COMMAND STATUS | (SHOW) | 7-52 |

PART 4. APPENDIXES

A. FORMAT PRESENTATION AND CODING RULES

| | | |
|------|---------------------------------|-----|
| A.1. | GENERAL RULES FOR THIS DOCUMENT | A-1 |
| A.2. | DATA DEFINITION CODING RULES | A-3 |

B. RESERVED WORDS

C. DATA DEFINITION PROCESSOR DIAGNOSTICS

D. UNIQUE LEXICON

E. DATA DEFINITIONS FOR UNIQUE EXAMPLES

INDEX

USER COMMENT SHEET

FIGURES

| | | |
|-------|--|------|
| 1-1. | Defined Record Management | 1-3 |
| 2-1. | Hierarchical Structure of a Defined File | 2-5 |
| 2-2. | Parent-Child Relationships in a Defined File | 2-5 |
| 2-3. | Fraternal Relationships in a Defined File | 2-5 |
| 3-1. | Overall Format of a Data Definition | 3-3 |
| 3-2. | Source Record Description Formats | 3-6 |
| 3-3. | Descriptions of Terms Used in the Definition Division | 3-7 |
| 3-4. | Consolidated Format of a Defined File Definition | 3-9 |
| 3-5. | Defined Record Definition Format | 3-12 |
| 3-6. | Describing the Defined Record's Source Using FROM Clause Format | 3-16 |
| 3-7. | Example Data Definition and UNIQUE Display Using FROM CONTROL BREAK | 3-19 |
| 3-8. | Sample Data Definition Using Repeating Group as Source of Child Record | 3-23 |
| 3-9. | Sample Data Definition Using Two Record Types as Sources of Parent and Child Records | 3-24 |
| 3-10. | Sample Data Definition Using Control Break as Source of Parent Record | 3-24 |
| 3-11. | Sample Data Definition Using Records in Different Indexed Files as Sources of Parent and Child Records | 3-25 |
| 3-12. | PREFIX Clause | 3-26 |
| 3-13. | POINTER Clause | 3-27 |
| 3-14. | Item Definition Format | 3-34 |
| 3-15. | Defined Record Identifier in a Simple Defined File | 3-35 |
| 3-16. | Terminal Display of Column Headers and Data Items in a Simple Defined File | 3-35 |
| 3-17. | Parent-Child Defined Record Identifiers | 3-36 |
| 3-18. | Terminal Displays of Column Headers and Data Items for Parent and Child Records | 3-37 |
| 3-19. | IDENTIFIER Statement Format | 3-37 |
| 3-20. | Terminal Display of Column Headers and Data Items for Multiple Identifiers | 3-39 |
| 3-21. | ITEM Statement Format | 3-41 |
| 3-22. | Supplement Definition Format | 3-48 |
| 3-23. | Deriving an Item from Another Source Record | 3-49 |
| 3-24. | Describing the Supplement's Source Using FROM Clause Format | 3-50 |
| 3-25. | POINTER Clause for Supplements | 3-53 |
| 3-26. | ITEM Definition Format for a Supplement | 3-62 |
| 3-27. | Subrecord Definition Format | 3-67 |
| 3-28. | Subitem Definition Format | 3-71 |
| 3-29. | Subfile Definition Format | 3-76 |
| 4-1. | Excerpt from EMPFILE, an Indexed Employee File | 4-3 |
| 4-2. | Data Definition for Defined File EMPLS | 4-3 |
| 4-3. | First Few EMPLS Records, as Listed at a Terminal by UNIQUE | 4-3 |
| 4-4. | First Few EMPLS Records, as Delivered to an Action Program | 4-3 |
| 4-5. | Action Program Descriptions of Defined Record EMPLOYEE | 4-3 |
| 4-6. | Excerpt from EMPFILE, an Indexed File with Two Record Types | 4-5 |
| 4-7. | Data Definition for Defined File EMPLOYEES | 4-5 |
| 4-8. | Excerpt from EMP-RG, an Indexed File Containing a Repeating Group | 4-6 |
| 4-9. | Data Definition for Defined File EMPLOYEES, Derived from a Repeating Group | 4-6 |
| 4-10. | Derivation of Defined File EMPLOYEES from Two Distinct Files Using Pointers | 4-7 |
| 4-11. | First Few EMPLOYEES Records, as Listed at a Terminal by UNIQUE | 4-9 |
| 4-12. | First Few EMPLOYEES Records, as Delivered to an Action Program | 4-9 |
| 4-13. | Action Program Descriptions of Defined Records EMPLOYEE and DEPENDENT | 4-9 |
| 4-14. | Indexed File with Two Records for Each Dependent (DEPFILE) | 4-11 |
| 4-15. | Excerpt from EMPFILE, an Indexed Employee File | 4-11 |

| | | |
|-------|---|------|
| 4-16. | Data Definition for Defined File DEPENDENTS Showing Supplements | 4-11 |
| 4-17. | First Few DEPENDENTS Records, as Listed at a Terminal by UNIQUE | 4-11 |
| 4-18. | First Few DEPENDENTS Records, as Delivered to an Action Program | 4-11 |
| 4-19. | Action Program Descriptions of Defined Record DEP-RECORD | 4-11 |
| 4-20. | Subfile Definition of SUBFIL, Restricting Access to Defined File EMPLS1 | 4-13 |
| 4-21. | First Few SUBFIL Records, as Listed at a Terminal by UNIQUE | 4-13 |
| 4-22. | First Few SUBFIL Records, as Delivered to an Action Program | 4-13 |
| 4-23. | Action Program Descriptions of Subrecord SUB-EMPS | 4-13 |
| 4-24. | Indexed Files EMPFILE and COFILE | 4-15 |
| 4-25. | Data Definitions for Related Defined Files EMPLOYEES and COMPANY | 4-15 |
| 4-26. | First Few EMPLOYEES Records, as Listed at a Terminal by UNIQUE | 4-15 |
| 4-27. | First Few COMPANY Records, as Listed at a Terminal by UNIQUE | 4-15 |
| 4-28. | First Few EMPLOYEES Records, as Delivered to an Action Program | 4-15 |
| 4-29. | First Few COMPANY Records, as Delivered to an Action Program | 4-15 |
| 5-1. | Data Definition Processing | 5-1 |
| 5-2. | Sample Job Control Streams to Execute the Data Definition Processor | 5-5 |
| 5-3. | COBOL Description of Defined File DEPENDENTS | 5-8 |
| 5-4. | Data Definition Processor Listings from Unsuccessful Runs | 5-10 |
| E-1. | Data Definition for Defined File INVFILE | E-1 |
| E-2. | Data Definition for Defined File SALES | E-2 |

TABLES

| | | |
|------|--|------|
| 3-1. | Data Definition Sections and Their Usages | 3-14 |
| 3-2. | Using POINTER and FILL KEY Clauses | 3-57 |
| 3-3. | ROLE IN UPDATE Options and Their Meanings | 3-60 |
| 7-1. | Symbols Used in SHOW Command Update Formats | 7-52 |
| B-1. | Reserved Words in the Definition Division | B-1 |
| B-2. | COBOL Reserved Words in the Data Division | B-2 |
| C-1. | Compilation Time Diagnostics Unique to the IMS Data Definition Processor | C-1 |
| D-1. | UNIQUE Commands | D-2 |
| D-2. | Punctuation Used in UNIQUE Commands | D-2 |
| D-3. | LIST Command Keywords | D-3 |
| D-4. | LIST Command Logical Operators | D-3 |
| D-5. | LIST Command Statistical Functions | D-3 |
| D-6. | Words Used in UNIQUE Status and Error Messages | D-4 |

PART 1. OVERVIEW



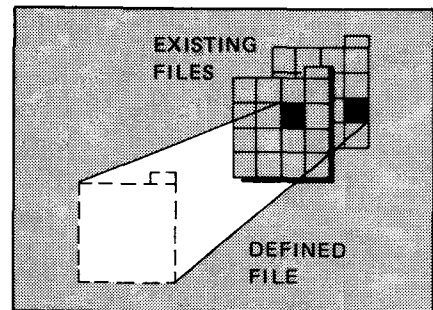
1. Introduction

1.1. CONCEPT AND PURPOSE

This manual tells you how to write data definitions to create information management system (IMS) defined files and how to use the IMS uniform inquiry update element (UNIQUE) to access those defined files.

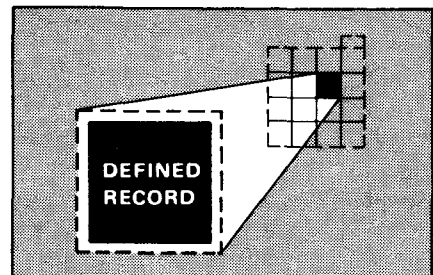
Defined file characteristics

A defined file is a logical file that IMS builds from records in your existing data files. Defined files exist only as descriptions in the named record (NAMEREC) file and need no additional storage.



Defined records

Defined files contain defined records.



UNIQUE requirements

UNIQUE accesses your data through defined files. UNIQUE is an easy-to-use inquiry language that lets you display data and update your files by entering commands from the terminal. A set of IMS-supplied action programs processes these UNIQUE commands.

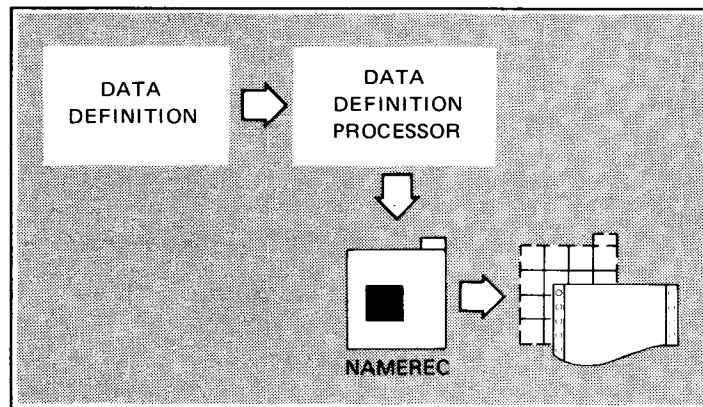
INTRODUCTION TO DATA DEFINITIONS**Action program requirements**

Your COBOL, basic assembly language (BAL), and RPG II action programs can access defined files, but this is not a requirement. Action programs can also access your existing conventional files. To use defined files with your action programs, you should also read the current version of the IMS action programming in COBOL and basic assembly language (BAL) user guide, UP-9207, or the IMS action programming in RPG II user guide, UP-9206.

1.2. CREATING DEFINED FILES**Creating defined files with the data definition processor**

You create a defined file by writing a data definition, using the data definition language (Section 3), and submitting it to an IMS utility program called the data definition processor (Section 5). The data definition processor:

- creates a data definition record in the NAMEREC file (see 2.3); and
- produces a printed description of the defined file and a diagnostic listing.

**Data definition elements**

In the data definition, you describe the structure of the defined file and defined records. You also specify the allowable updating functions (modify, add, delete) and value ranges.

Requesting defined records

When an action program or a terminal operator using UNIQUE requests a record, IMS constructs the defined record and passes it to the action program or UNIQUE.

1.3. ACCESSING DEFINED FILES THROUGH UNIQUE

*Function of
UNIQUE*

UNIQUE uses information you supply in your data definition to format output screens, restrict updating, and validate entries from terminals.

UNIQUE commands

You can use UNIQUE commands to:

- Display a record
- Add, delete, or change a record
- List all or selected portions of a file
- Obtain statistical data about a file

Section 7 contains descriptions and extended examples of UNIQUE commands.

1.4. DEFINED RECORD MANAGEMENT

Characteristics

Defined record management (Figure 1-1) is the IMS component handling requests from UNIQUE and action programs for defined records.

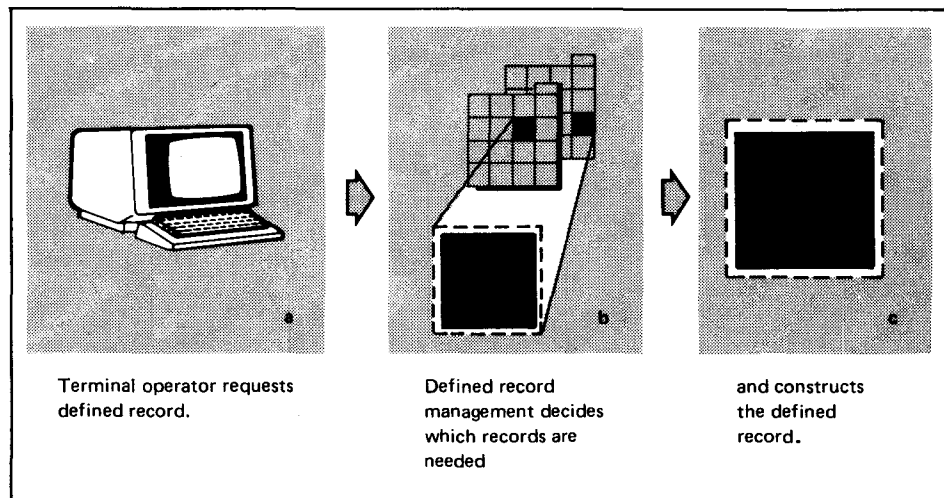


Figure 1-1. Defined Record Management

INTRODUCTION TO DATA DEFINITIONS

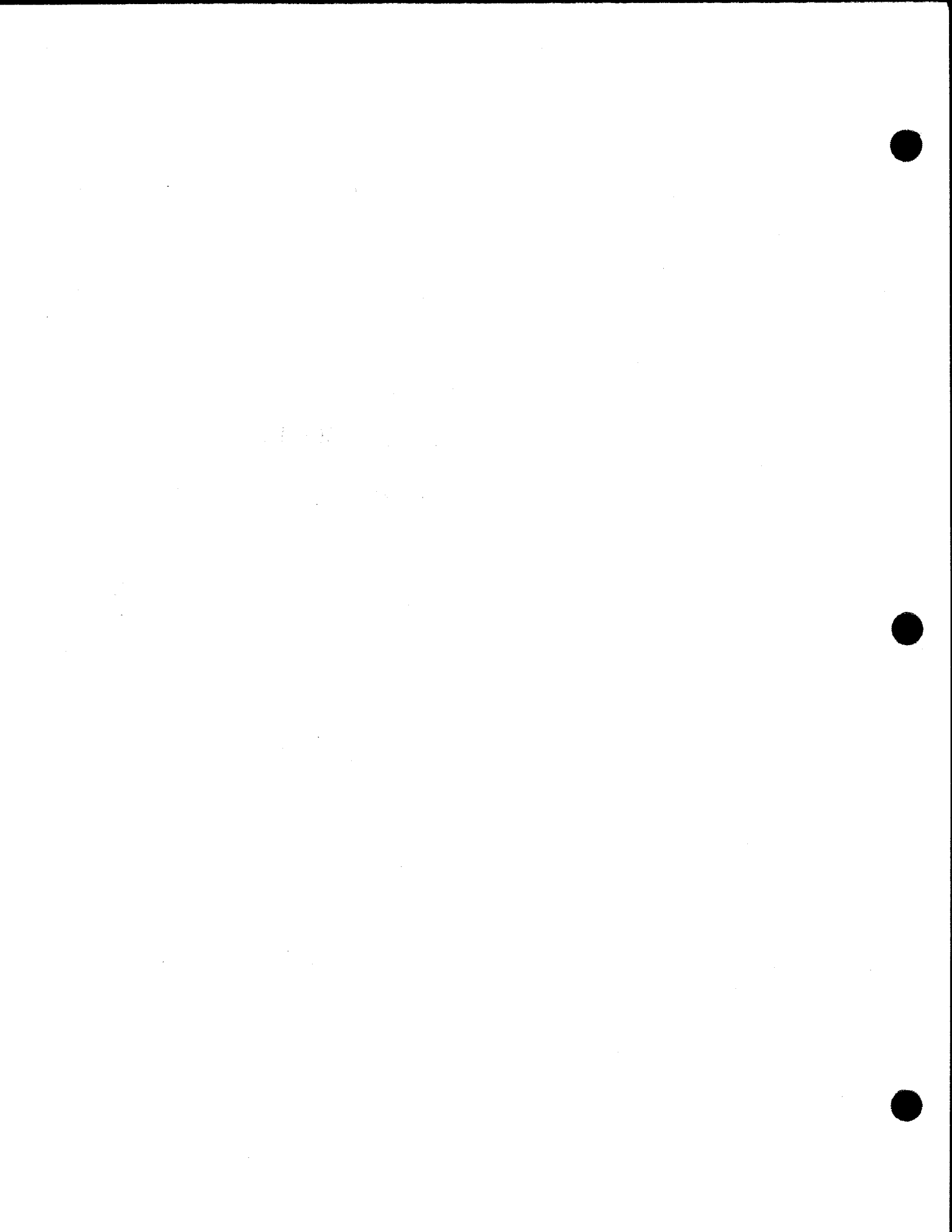
Constructing defined records

When a terminal operator requests a defined record, defined record management decides which physical records or parts of them are needed and which files they are located in. This information comes from the data definition record in the NAMEREC file. Defined record management constructs the defined record and passes it to UNIQUE or the action program.

Validating updates

When you try to update a file in your action program or with UNIQUE, defined record management checks that the changes are permitted and that the values are within the limits specified in your data definition.

PART 2. DATA DEFINITIONS

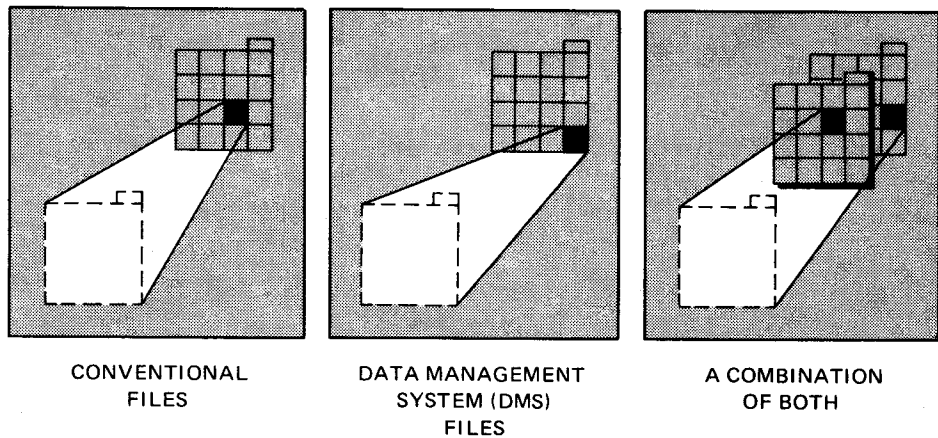


2. Defined File Structure

2.1. DEFINED FILES

Defined file sources

Defined files are built from records in:

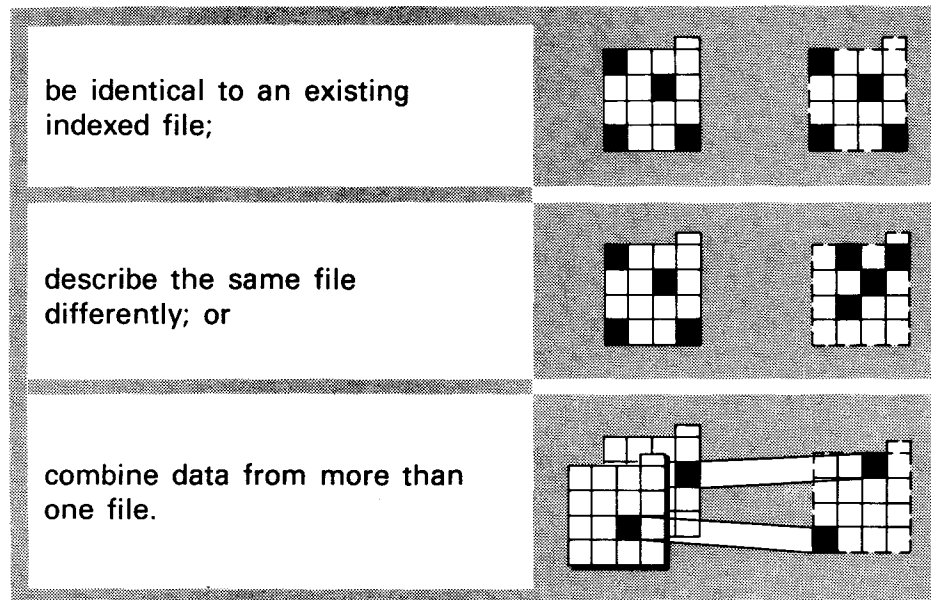


Building defined files

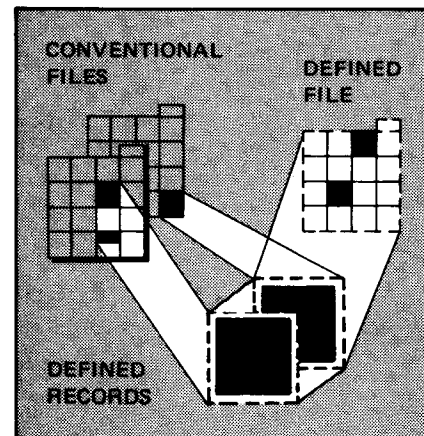
Because identifiers (3.19) are taken from keys, you can only build defined files from indexed files (indexed sequential access method (ISAM) or multiple indexed random access method (MIRAM)) or a database subschema. You can only use nonindexed files (direct access method (DAM) or MIRAM) when they are combined with indexed files or a subschema.

DEFINED FILE STRUCTURE*Defined file structure*

Defined files take many forms. They can:

**2.2. DEFINED RECORDS***Characteristics*

Defined records make up a defined file. They redefine records in your existing files and contain the data you need for an application.

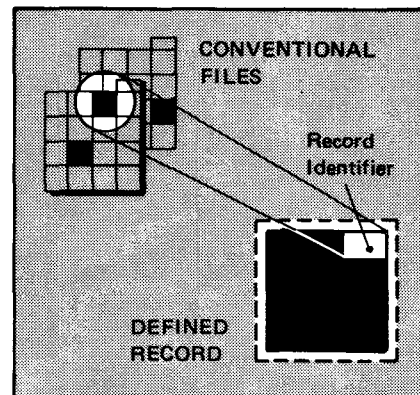
*Defined record sources*

A defined record consists of:

- all or part of a record from one file;
- all or parts of several records from the same file; or
- all or parts of several records from different files.

Identifiers

Defined records contain record identifiers, which come from record keys. These identifiers locate the data in your conventional or data base files that make up your defined records. See 3.19 through 3.22 for a detailed explanation of defined record identifiers.

**Supplements**

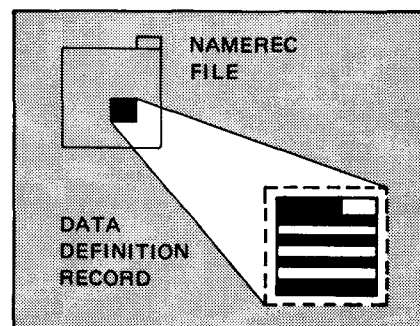
Defined records can contain additional items taken from different fields in the same record or from other records. You need to write supplement definitions to include these items in a defined record. (See 3.30 through 3.41.)

2.3. DATA DEFINITION RECORDS**Contents of data definition record**

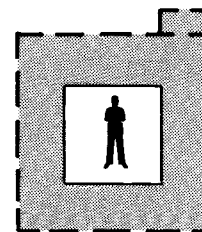
A data definition record contains a description of the defined file and its related subfiles (3.53). IMS uses this information to construct defined records requested by action programs and UNIQUE.

Creating and storing data definition records

The data definition processor creates a data definition record from the data definition you write. Data definition records are stored in the NAMEREC file. This internal file holds records and tables needed by IMS during online operations.

**2.4. STRUCTURE OF DEFINED FILES****Characteristics**

A defined file containing one record type is a simple defined file.



DEFINED FILE STRUCTURE

A defined file containing more than one type of defined record has a hierarchical structure. Records in a hierarchical defined file have parent, child, and fraternal relationships.

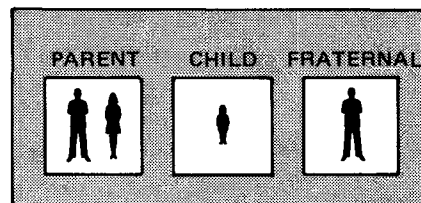


Figure 2-1 shows the hierarchical structure of a defined file. In practice, most defined files contain few types of defined records; this example contains many for illustration.

The parent-child relationships are:

| <i>Parent-child records</i> | <u>Parent Record</u> | <u>Child Records</u> |
|-----------------------------|----------------------|----------------------|
| | A1 | B1, B2, B3 |
| | B1 | C1, C2, C3 |
| | B3 | C4, C5 |
| | C4 | D1 |

Fraternal records

Fraternal records are at the same level in the hierarchy. They have the same parent or no parent.

The fraternal records are:

Set 1: A1, A2

Set 2: B1, B2, B3

Set 3: C1, C2, C3

Set 4: C4, C5

C1, C2, and C3 are not fraternal to C4 and C5 because they have different parents.

Hierarchical record order

In your data definition, you must define parent, child, and fraternal records in a specific order. They appear in that same order in the defined file. A parent record is defined first, followed by each of its child records. Each of these child records is followed by any child records to which it is a parent.

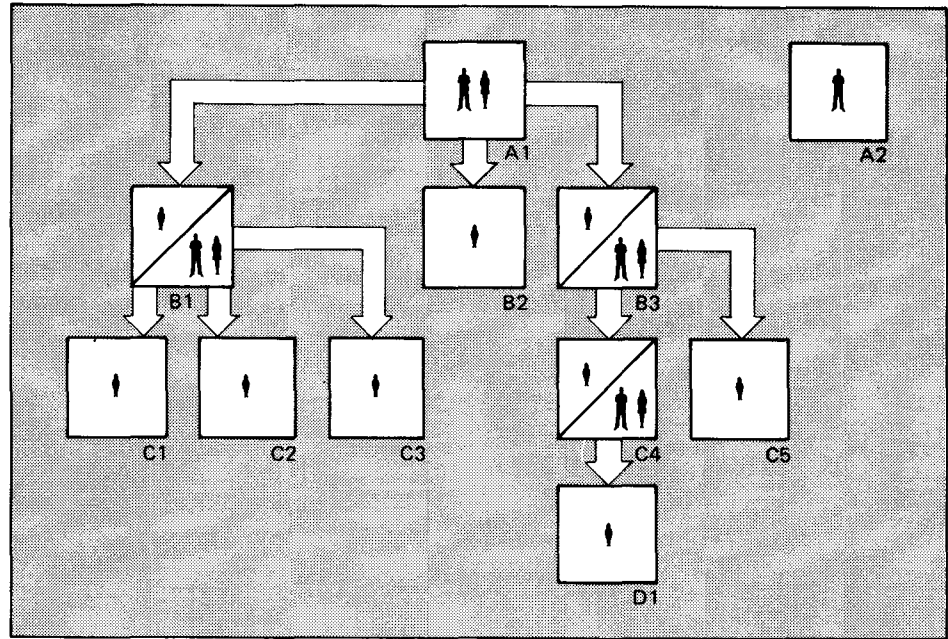


Figure 2-1. Hierarchical Structure of a Defined File

Examples of hierarchical records

Figures 2-2 and 2-3 show the order in which the defined records in Figure 2-1 are defined. Figure 2-2 also shows the parent-child relationships in the defined file, and Figure 2-3 shows the fraternal relationships.

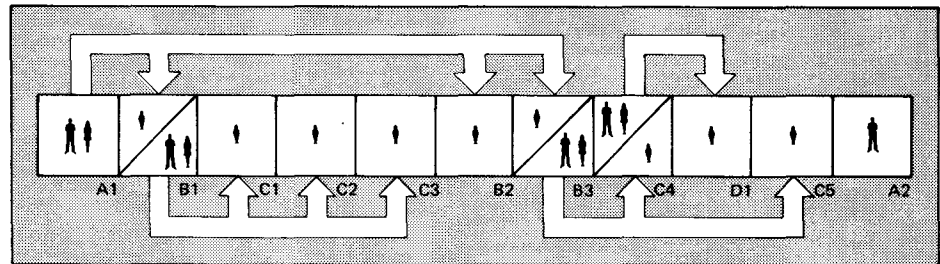


Figure 2-2. Parent-Child Relationships in a Defined File

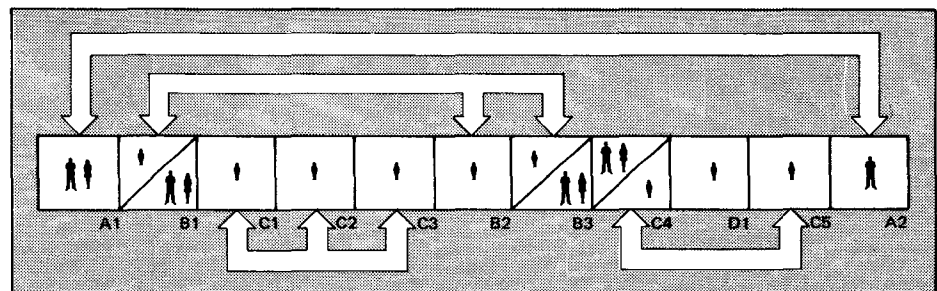


Figure 2-3. Fraternal Relationships in a Defined File



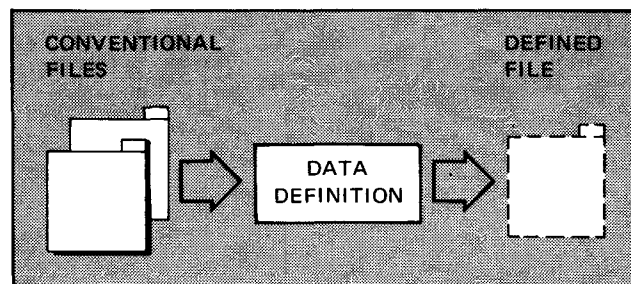
3. Writing Data Definitions

3.1. DATA DEFINITION LANGUAGE

Characteristics

Sources

The data definition language is used to describe the defined files accessed by your action programs or UNIQUE. It is similar to COBOL. Each data definition describes one defined file in terms of one or more indexed files, or a combination of indexed and nonindexed files. You can create many defined files through multiple runs of the data definition processor.



You can also use DMS data base subschema as a source in a data definition. The IMS/DMS interface user guide, UP-8748 (current version) describes the language for a data definition using subschema records.

Conventions and formats

Appendix A gives the statement conventions and the format presentation and coding rules for the data definition. Appendix B lists reserved words you cannot use in the definition division.

DATA DEFINITION LANGUAGE

*Using Katakana
characters***NOTE:**

In your data definition, you can use Katakana characters for:

- *Defined file and subfile names*
- *Defined record names*
- *Identifier names*
- *Item names*

To use the Katakana character set, you must specify KATAKANA=YES in the network section of the IMS configuration. For details, see the IMS system support functions user guide, UP-8364 (current version).

3.2. DATA DEFINITION STRUCTURE

*Overall structure –
three divisions*

Figure 3-1 shows the overall structure of the data definition. It contains: an identification division, a data division, and a definition division.

*Similarity to
COBOL*

The identification and data divisions are similar to COBOL. While the definition division is unique to IMS, its syntax is very similar to COBOL. The data division describes files that the defined file is extracted from; the definition division describes the defined file.

The *record-description* and *defined-file-definition* group formats are expanded in Figures 3-2 and 3-3.

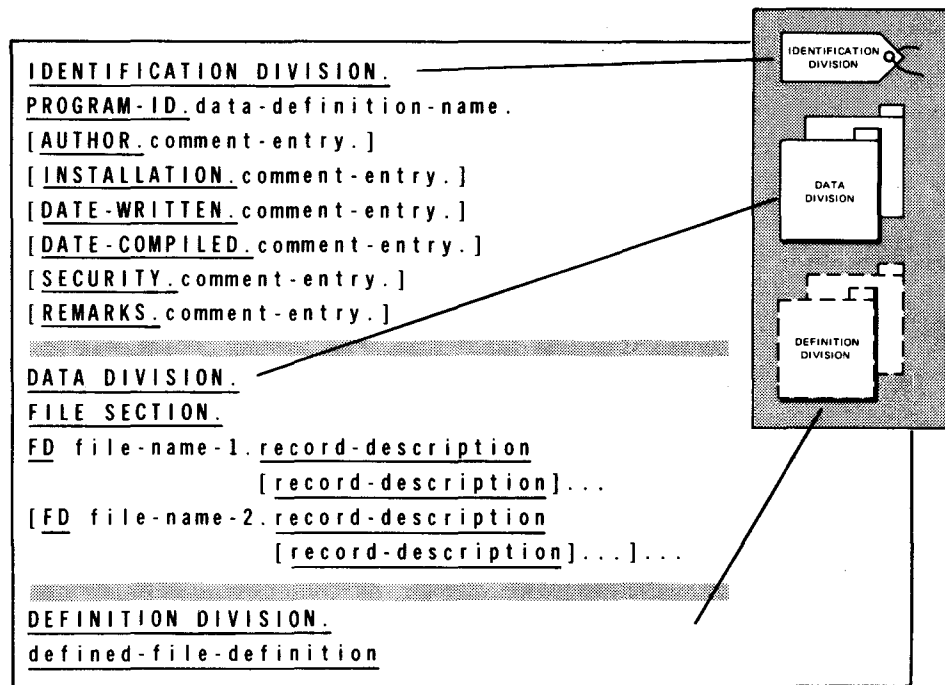


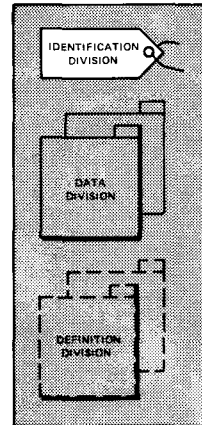
Figure 3-1. Overall Format of a Data Definition

DATA DEFINITION STRUCTURE

3.3. IDENTIFICATION DIVISION

Required entries Begin the identification division with the reserved words IDENTIFICATION DIVISION. Next, give the PROGRAM-ID statement.

Data-definition-name A data-definition-name follows the PROGRAM-ID header. This becomes the name of the program; it appears on the output listing and identifies the contents of the listing. It is alphanumeric, beginning with an alphabetic character. In the compiler listing, only the first six characters are printed out to identify the program. You can use more characters, but it is difficult to identify your programs when the first six characters are not unique. Each statement begins in margin A (column 8) of the coding form.



Optional entries Each optional comment entry (AUTHOR, INSTALLATION, DATE-WRITTEN, DATE-COMPILED, SECURITY, and REMARKS) can contain any printable characters. When an entry exceeds a single line, begin additional lines in column 12 or beyond.

Example

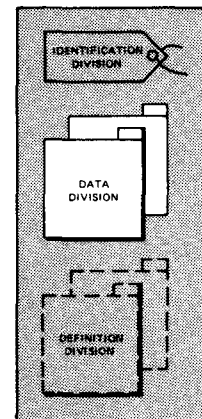
```

8  12
IDENTIFICATION DIVISION.
PROGRAM-ID. PAYROLL1.
AUTHOR. JOHN SMITH.

```

3.4. DATA DIVISION

Contents The data division contains only a file section, which describes your conventional files. Although it is similar to COBOL, it cannot contain the VALUE clause. Other clauses are the same as in COBOL.



Required entries Begin with the reserved words DATA DIVISION and FILE SECTION. DATA DIVISION, FILE SECTION, FD statements, and 01-level record descriptions begin in column 8. Start all other entries in column 12 or beyond.

FD statements FD statements describe records in conventional files that the defined file is extracted from. Filename-1, filename-2, etc, identify the conventional files and begin in column 12 or beyond. Use the same file names in the filename positional parameters in the configurator FILE section. (See the IMS system support functions user guide, UP-8364 (current version).)

Record-descriptions Record-description entries describe the source records. Figure 3-2 shows two formats, both similar to COBOL.

Source file descriptions You can describe more than one source file, each containing multiple record descriptions.

Example

```

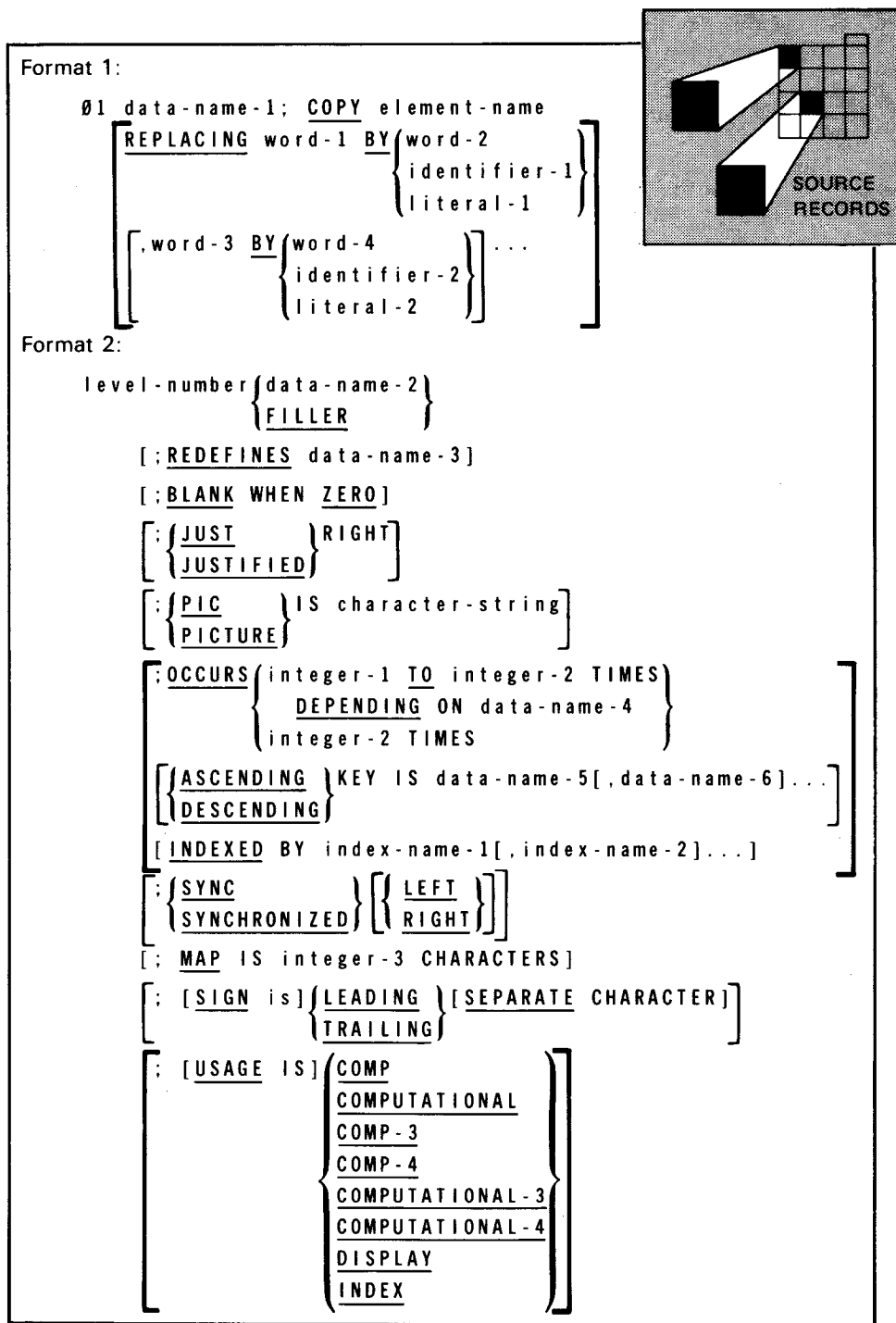
8  12
DATA DIVISION.
FILE SECTION.
FD  EMPFILE.
01  EMP-REC.
     02  EMP-NAME           PIC X(21).
     02  EMP-NO             PIC X(5).
     02  FILLER             PIC X(7).
FD  DEPFILE.
01  DEP-REC.
     02  DEP-NAME           PIC X(21).

```

UNIQUE requirement**NOTE:**

If you plan to use any UNIQUE statistical functions for a data item, define that item as numeric.

DATA DEFINITION STRUCTURE



NOTE:

The DEPENDING ON option of the OCCURS clause is ignored by the data definition processor.

Figure 3-2. Source Record Description Formats. Format 1 copies data descriptions from an existing library; format 2 further describes the record fields given in format 1.

3.5. DEFINITION DIVISION

Contents

The definition division describes the defined file. In the definition division, you name a defined file and describe each defined record, item, supplement, subrecord, subitem, and subfile (3.53). See Figure 3-3.

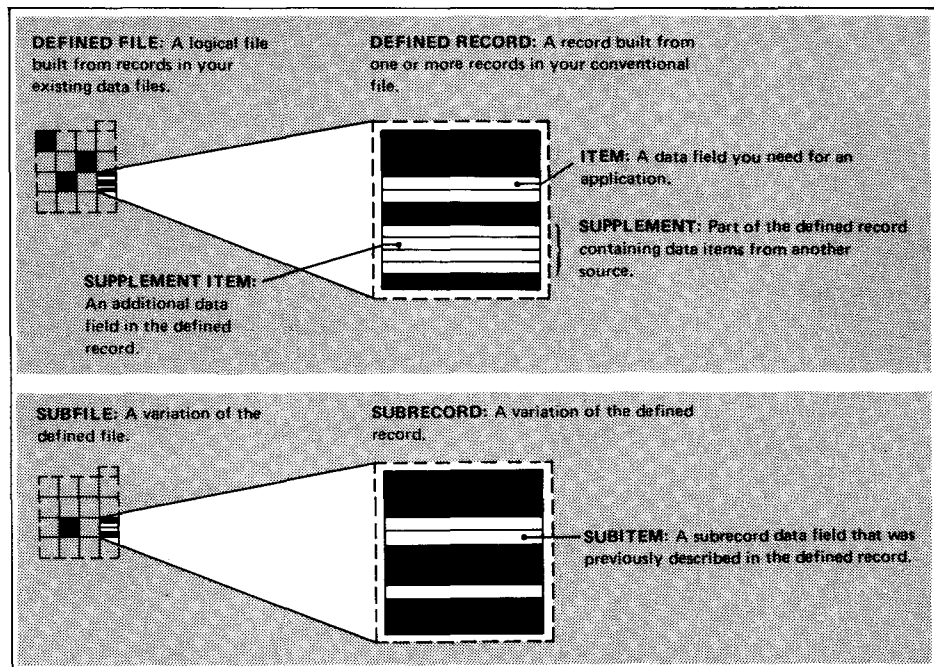
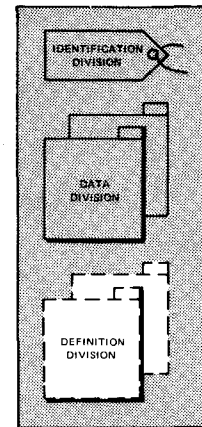


Figure 3-3. Descriptions of Terms Used in the Definition Division

Begin the definition division in column 8 with the reserved words DEFINITION DIVISION. A description of the defined-file-definition follows in 3.6.

Example

8 12
DEFINITION DIVISION.

DEFINED FILE DEFINITION**3.6. DEFINED FILE DEFINITION****Contents**

The defined file definition contains the defined record, item, supplement, subrecord, and subfile definitions. IMS uses them to construct the defined file from your conventional files.

Only one defined file definition

Each data definition can define only a single defined file.

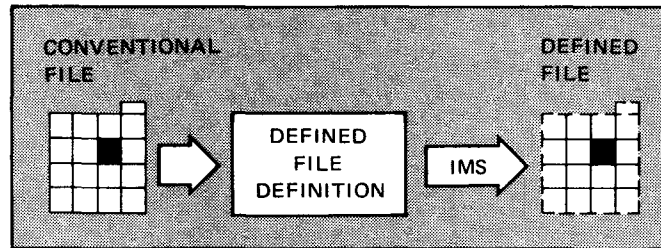
**Required and optional entries**

Figure 3-4 shows the consolidated format for the defined file definition. Statements enclosed in solid-line boxes are required; those in broken-line boxes are optional. You must include statements in the inner solid-line boxes when you include the statement in the outer broken-line boxes.

Nested structure

In this nested structure (boxes within boxes), the defined record and subfile definitions are subordinate to the defined file definition; the item, supplement, and subrecord definitions are subordinate to the defined record definition. You can use all subordinate definitions repeatedly within the larger definitions.

Defined record positions

You can define multiple defined record types in the defined file definition. The positions of the defined record types within the defined file match the order in which you define them.

The boxed statements in Figure 3-4 are described in 3.7 through 3.55.

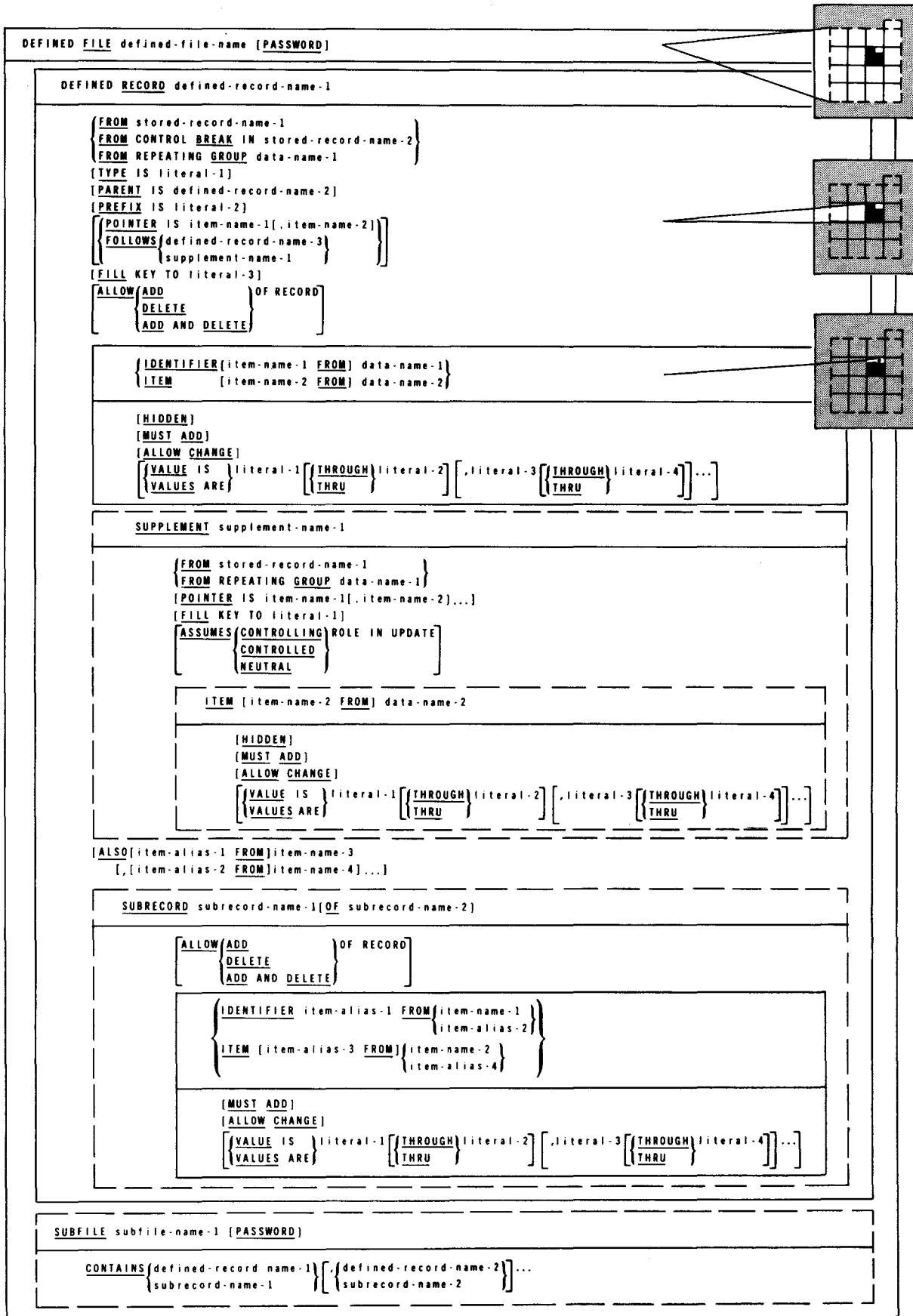
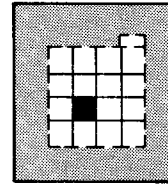


Figure 3-4. Consolidated Format of Defined File Definition

DEFINED FILE DEFINITION**3.7. NAMING THE DEFINED FILE (DEFINED FILE STATEMENT)****Function**

The defined file statement begins a defined file definition and names the file. It also creates a record key for the data definition record. Within the named record (NAMEREC) file, each defined file name must be unique. The format is:

**Format**

`DEFINED FILE defined-file-name [PASSWORD]`

Coding rule

This is the first statement in the definition division and begins in column 8.

Defined-file-name

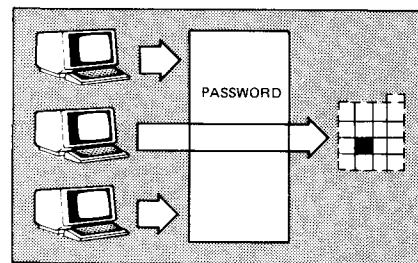
The defined-file-name is one to seven characters and must differ from the names of any conventional files assigned to IMS. The data definition processor truncates names longer than seven characters, but it does not issue any error message.

Passwords**PASSWORD clause**

Passwords protect defined files by limiting access to them. With UNIQUE, you can use the PASSWORD clause; with action programs, you cannot. When you specify PASSWORD, terminal operators enter the defined-file-name as a password in the UNIQUE OPEN command (7.3) to access a defined file.

**Defining passwords
with the NAMEREC utility**

You can omit PASSWORD and define a password with the NAMEREC file utility. This allows you to limit defined file access to specific UNIQUE terminals and use multiple passwords to access the same defined file. A password defined in the NAMEREC utility does not cancel one defined in the data definition unless the passwords are the same. The IMS system support functions user guide, UP-8364 (current version) describes password definition with the NAMEREC utility.



Effect of omitting password definition

You must define a password using either the PASSWORD clause or the NAMEREC utility; otherwise, terminal operators using UNIQUE cannot access the defined file.

Outside references

You use the defined file name to refer to the defined file in a number of places outside the data definition:

- Keyword parameters DFILE and DDRECORD in the ACTION section of the configuration
- Keyword parameters FN and DDN in the password definition input to the NAMEREC file utility
- The defined-file-name parameter in action program function calls to defined record management
- The defined-file-name and data-def-rec-name fields in the program information block for COBOL, BAL, and RPG II action programs

The IMS system support functions user guide, UP-8364 (current version) describes IMS configuration and the NAMEREC file utility. Action programs are discussed in the current version of the IMS action programming in COBOL and BAL user guide, UP-9207 and the IMS action programming in RPG II user guide, UP-9206.

Examples

```
      8  12  
      _____  
      DEFINITION DIVISION.  
1.  DEFINED FILE EMPFILE PASSWORD  
2.  FILE EMPFILE PASSWORD  
3.  FILE EMPFILE
```

Examples 1 and 2 perform exactly the same function. The defined file name is EMPFILE, which is also the password used to access the file through UNIQUE. PASSWORD is omitted in example 3, preventing file access when using UNIQUE. In this case, either the NAMEREC utility creates a password, or only your action programs, not UNIQUE, access the defined file.

DEFINED RECORD DEFINITION**3.8. DEFINED RECORD DEFINITION***Function*

A defined record definition describes the sources and contents of each defined record and allowable updating functions. Write a separate defined record definition for each defined record type in the defined file. Figure 3-5 shows the format of the defined record definition. Underlined lowercase terms are group formats described in separate subsections (3.19 through 3.52).

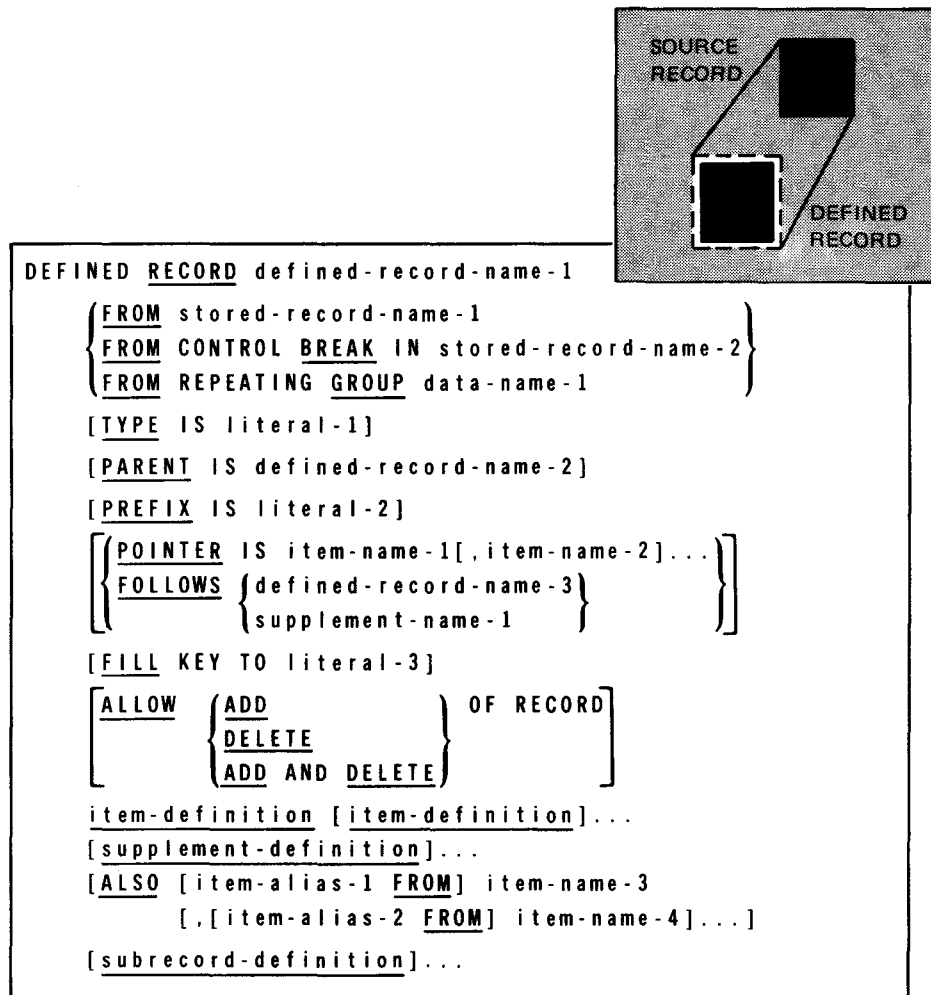
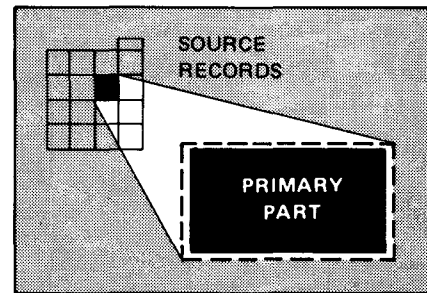
Format

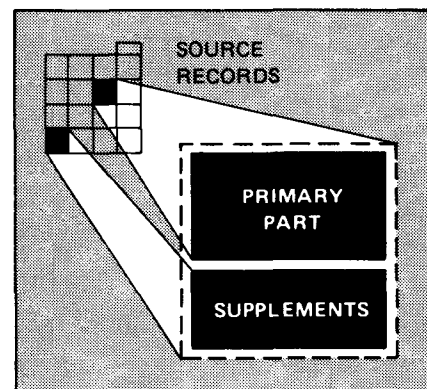
Figure 3-5. Defined Record Definition Format

Primary part

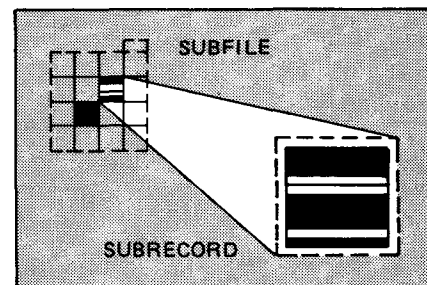
A defined record definition always contains a DEFINED RECORD statement and one or more item definitions. These statements define the primary part of the defined record. The source of this part is always an indexed file or a data base subschema. The DEFINED RECORD statement is described in 3.9 through 3.18 and the item definition in 3.19 through 3.29.

**Supplements**

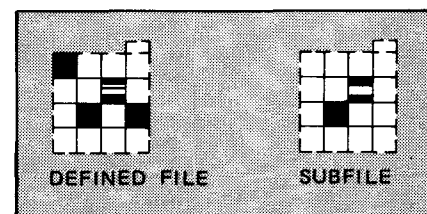
When the defined record contains additional items from the same or another source record, the defined record has a primary part and one or more supplements. You define the source and contents of each supplement in a supplement definition. The supplement definition is described in 3.30 through 3.42.

**Subrecords**

When you want to describe a variation of the defined record, you include a subrecord definition. A subrecord contains the same data as in the defined record, but the data may be in a different order or the allowable updating functions may vary. The subrecord definition is described in 3.43 through 3.52.

**Subfiles**

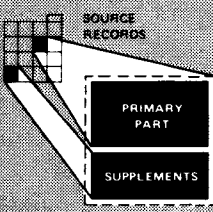
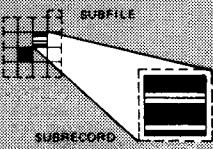

When you want to describe a variation of the defined file, you include a subfile definition. A subfile definition describes a subset of a defined file and is used to access subrecords. It can differ from the defined file in the number and makeup of the defined record types it contains. The subfile definition is described in 3.53 through 3.55.



DEFINED RECORD DEFINITION

Table 3-1 summarizes the sections of the data definition and their usages.

Table 3-1. Data Definition Sections and Their Usages

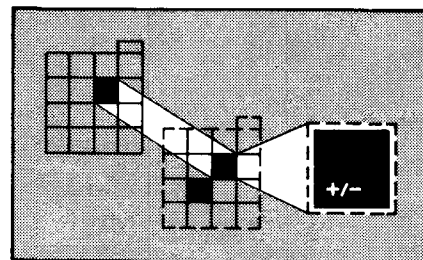
| Use a: | When you want: |
|--|--|
| <p>SUPPLEMENT</p>  | <p>Your defined record to include additional items from the same or another source record.</p> |
| <p>SUBRECORD</p>  | <p>To describe a variation of your defined record.</p> |
| <p>SUBFILE</p>  | <p>To describe a variation of your defined file.</p> |

3.9. DESCRIBING A DEFINED RECORD'S PRIMARY PART (DEFINED RECORD STATEMENT)

Function

The DEFINED RECORD statement describes:

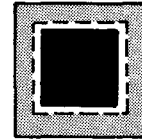
- the source of the primary part of the defined record;
- the defined record's relationship to other records in the defined file; and
- whether terminal operators using UNIQUE or action programs may add or delete occurrences of this record.



3.10. NAMING THE DEFINED RECORD (DEFINED RECORD CLAUSE)

Function

The **DEFINED RECORD** clause begins a defined record definition and names the defined record. Put it directly after the **DEFINED FILE** statement or another defined record definition. Starting in column 8, the format is:



Coding rule

Format

DEFINED RECORD defined-record-name-1

Defined-record-name-1

Defined-record-name-1 is a 1- to 30-character name, unique within the data definition, identifying the defined record.

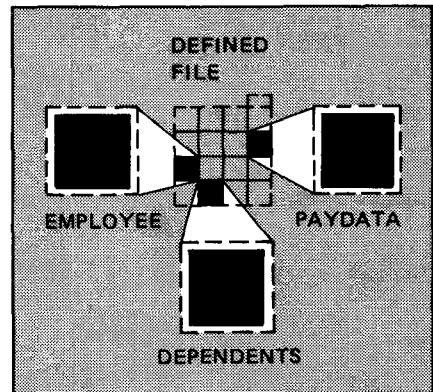
Examples

```

      8  12
1.  DEFINED FILE PAYROLL
    DEFINED RECORD EMPLOYEE
2.  DEFINED FILE PAYROLL
    RECORD EMPLOYEE
3.  DEFINED FILE PAYROLL
    DEFINED RECORD EMPLOYEE
    .
    .
    DEFINED RECORD DEPENDENTS
    .
    .
    DEFINED RECORD PAYDATA

```

Examples 1 and 2 are the same; **DEFINED** is not required. **EMPLOYEE** is a defined record in defined file **PAYROLL**. In example 3, defined file **PAYROLL** contains three defined records: **EMPLOYEE**, **DEPENDENTS**, and **PAYDATA**.



DEFINED RECORD DEFINITION

3.11. IDENTIFYING THE SOURCE OF THE PRIMARY PART (FROM CLAUSE)

Function

The FROM clause specifies the source of the primary part of the defined record. Place it directly after the DEFINED RECORD clause.

Three formats

The FROM clause has three formats. Figure 3-6 shows the use of these formats.

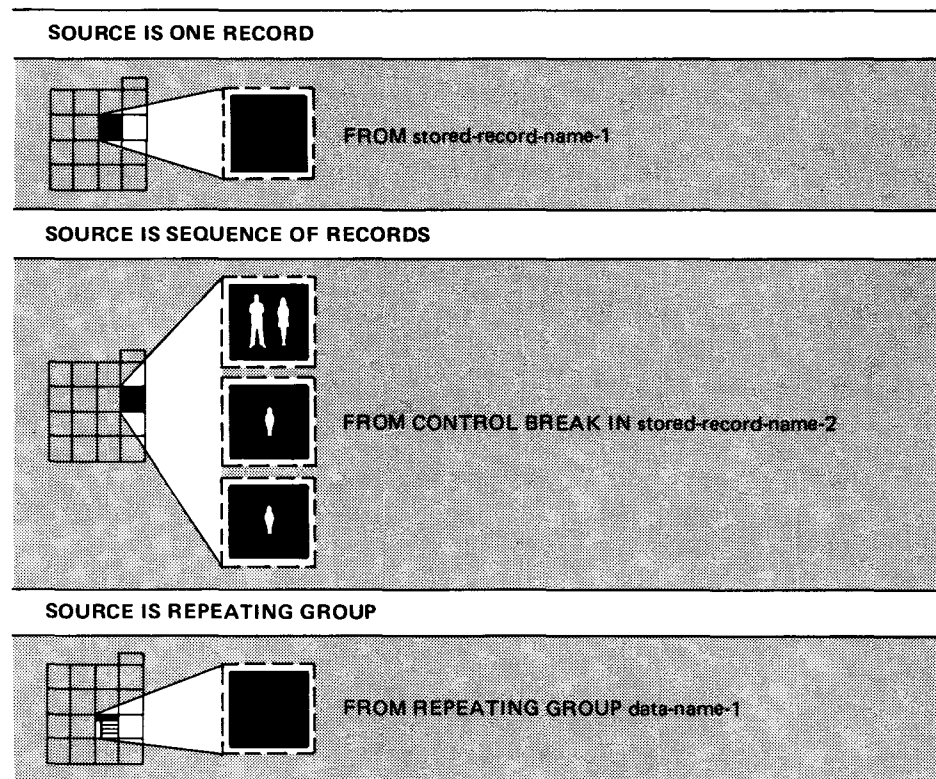


Figure 3-6. Describing the Defined Record's Source Using FROM Clause Format

Specifying One Record as the Source

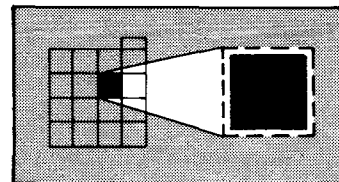
Use the format

Format

FROM stored-record-name-1

Purpose

when the source of the primary part of the defined record is one record. This source record must be in an indexed file. The defined record's primary part contains the record identifier (derived from the record key) and any items in the same record.



Stored-record-name-1

Stored-record-name-1 refers to an 01-level record description in the data division (3.4). The name must be unique. This defined record's primary part can include a data item from this source record when:

- the data item meets length and usage constraints (3.24); and
- within stored-record-name-1, the data item precedes any item defined with an OCCURS clause (see Figure 3-2).

Example

```
8 12
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
```

In this example, the primary part of defined record EMPLOYEE comes from record EMPLOYEE-REC. EMPLOYEE-REC is an 01-level record description.

Specifying a Sequence of Records as the Source

Use the format

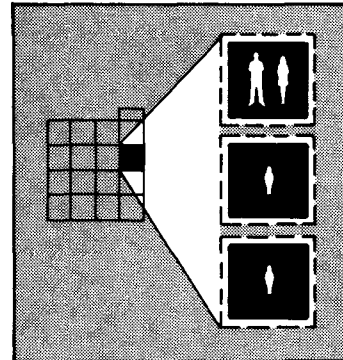
Format

```
FROM CONTROL BREAK IN stored-record-name-2
```

when:

Purpose

- you have parent and child defined records; and
- the primary part of the parent defined record comes from a sequence of records with the same leftmost values in their record keys.

**How control breaks work**

The parent defined record's primary part contains only an identifier. Other items are contained in subordinate (child) defined records, which name the same source record (stored-record-name-2). As indexed records are read sequentially, each value change in the left-hand character positions of the identifier record key produces a new occurrence of the current defined record.

DEFINED RECORD DEFINITION

Uses Use this format to access a specific portion of a defined file with, for example, the FOR parameter of a UNIQUE LIST or DETAIL command. It also enables UNIQUE statistical functions to provide subtotals for child defined record subsets associated with control breaks.

Stored-record-name-2 Stored-record-name-2 is an 01-level record description in the data division (3.4). The name must be unique.

Examples

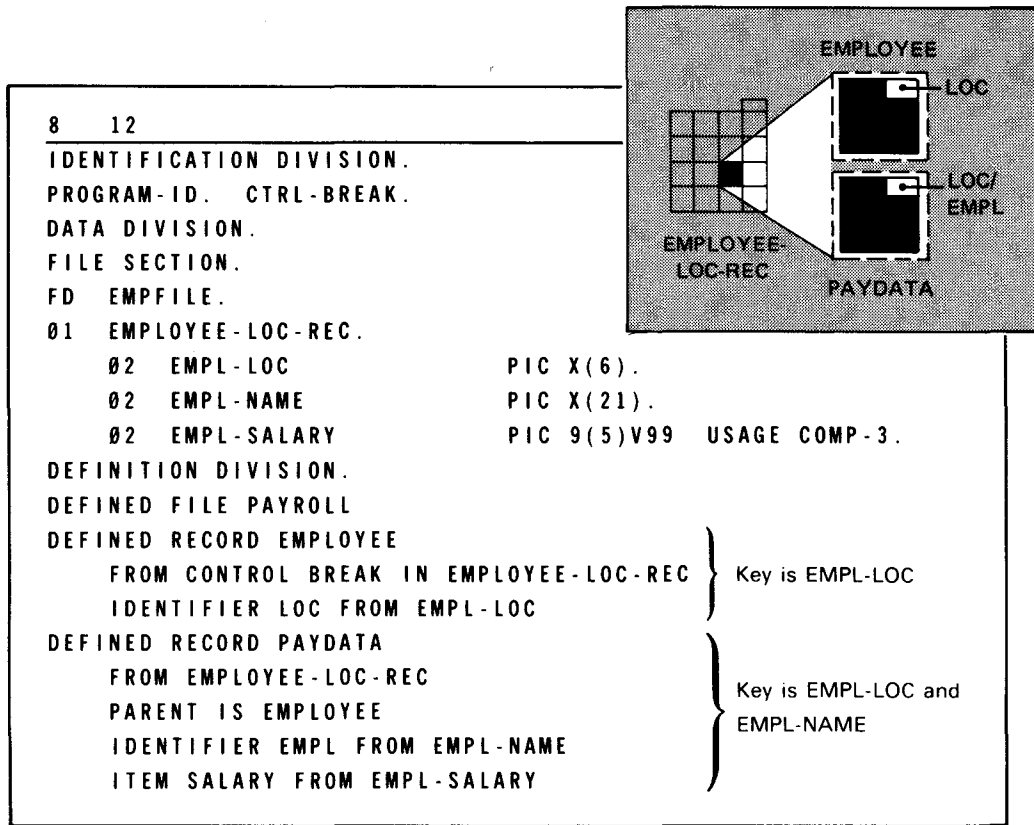
| | | |
|----|---|----|
| | 8 | 12 |
| 1. | DEFINED RECORD EMPLOYEE | |
| | FROM CONTROL BREAK IN EMPLOYEE-LOC-REC | |
| 2. | RECORD EMPLOYEE FROM BREAK EMPLOYEE-LOC-REC | |

Example 1 uses the long form of the DEFINED RECORD and FROM CONTROL BREAK statements; example 2 uses the short form. Both perform the same function. Defined record EMPLOYEE receives only an identifier from logical record EMPLOYEE-REC, which is an 01-level record description.

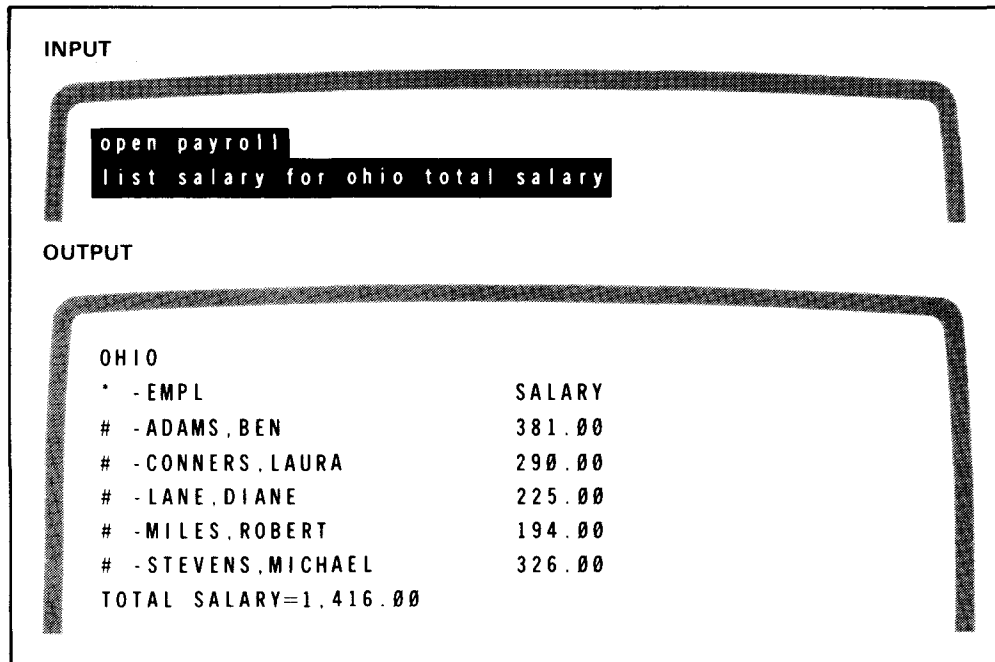
Figure 3-7 shows a sample data definition for defined file PAYROLL, the UNIQUE commands used to access it, and the resulting display. IDENTIFIER and ITEM statements are discussed in 3.20 through 3.28.

NOTE:

In the screen displays in this manual, the entries you make are shown in lowercase and reverse print.



a. Data definition



b. UNIQUE display

Figure 3-7. Example Data Definition and UNIQUE Display Using FROM CONTROL BREAK. Defined record PAYDATA names EMPLOYEE as its parent and EMPLOYEE-REC as its source. With FROM CONTROL BREAK, you can restrict the listing to employee names and salaries for the Ohio office.

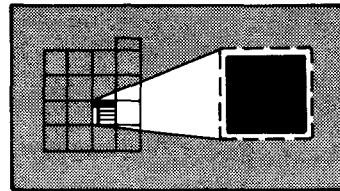
DEFINED RECORD DEFINITION

Specifying a Repeating Group as the Source

Use the format

Format FROM REPEATING GROUP data-name-1

Purpose when the source of the primary part of the defined record is a group item. This group item is described in the data division with an OCCURS clause.



Data-name-1 Data-name-1 is a data name defined in the data division with both OCCURS and KEY clauses. (See Figure 3-2.) It must be unique or fully qualified. Include any data-name-1 item in the primary part of this defined record when:

- the item meets length and usage constraints (3.24); and
- within data-name-1, it precedes any item (other than data-name-1 itself) defined with an OCCURS clause.

Restrictions on adding records Do not use this format when you want to add records with a UNIQUE ADD command, INSERT function, or ADD specification in an action program. Adding a record produces binary zeros as the value of data-name-1, so it cannot contain a unique key.

Example

```

8  12
DATA DIVISION.
FILE SECTION.
FD  EMPFILE.
01  EMPLOYEE-REC.
    02  EMPLOYEE                PIC X(21).
    02  DEPENDENT-REC OCCURS 5 TIMES
        ASCENDING KEY IS DEP-NAME.
    03  DEP-NAME                PIC X(21).
DEFINITION DIVISION.
DEFINED FILE PAYROLL PASSWORD
RECORD EMPLOYEES FROM EMPLOYEE-REC
.
.
RECORD DEPENDENTS FROM GROUP DEPENDENT-REC

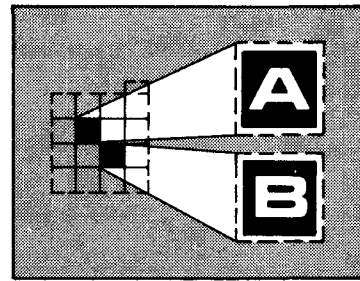
```

In this example, repeating group item DEPENDENT-REC (an 02-level entry in logical record EMPLOYEE-REC) supplies the primary part of defined record DEPENDENTS. DEPENDENT-REC must appear in the data division file section with both OCCURS and KEY clauses.

3.12. DEFINING THE RECORD TYPE (TYPE CLAUSE)

Purpose

Use the TYPE clause only with your action programs, not UNIQUE. With this clause, you can specify the record type you want delivered with a SETL and sequential GET function. The detailed status code field in the program information block (PIB) gives the type indicator. Use this statement only when a given defined file accessed by your action programs contains more than one record type.



Controlling record delivery

You can control the way the TYPE clause is used. Depending on the settings of the PREDICTED (byte 1 of the detailed status code) and DELIVERED (byte 2 of the detailed status code) indicators in the program information block, the record is:

- retrieved and then checked for the TYPE; or
- checked for the TYPE and then retrieved.

The format is:

Format

TYPE IS literal-1

Literal-1

Literal-1 is the actual value associated with the record type delivered in the program information block's detailed status code field. (See the current version of the IMS action programming in RPG II user guide, UP-9206 or the IMS action programming in COBOL and basic assembly language (BAL) user guide, UP-9207 for details.) Use one alphanumeric character and assign a unique character identification to each defined record type.

Example

```

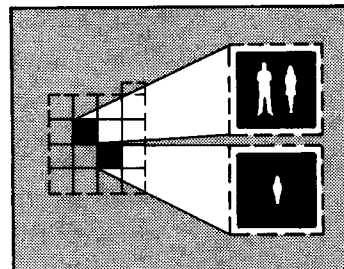
8 12
DEFINED RECORD EMPLOYEE-SALARY FROM EMPLOYEE-REC
    TYPE IS 'A'
DEFINED RECORD EMPLOYEE-HISTORY FROM DEPT-REC
    TYPE IS 'B'

```

In this example, the TYPE clause indicates that a SETL and sequential GET function delivers an 'A' record type for record EMPLOYEE-SALARY. Record EMPLOYEE-HISTORY, located in the same defined file, is a 'B' record type.

DEFINED RECORD DEFINITION**3.13. IDENTIFYING THE PARENT OF A CHILD DEFINED RECORD
(PARENT CLAUSE)***Purpose*

With the PARENT clause, you set up the hierarchical relationship between defined records within the defined file.



The format is:

Format

PARENT IS defined-record-name-2

Defined-record-name-2

Defined-record-name-2 is:

- a record defined in the immediately preceding defined record definition; or
- a direct ancestor of the immediately preceding defined record.

Hierarchical relationships

Every defined record definition but those at the highest level in the hierarchy must contain a PARENT statement. The first defined record definition is at the highest level in the hierarchy; it does not have a parent record. When a subsequent defined record definition has no PARENT statement, it is also considered to be at the highest hierarchical level.

Fraternal records

All record types having no parents are fraternal, as are defined records naming the same parent. (See 2.4.)

Example

```
8 12
RECORD EMPLOYEE FROM EMPLOYEE-REC
```

```
RECORD PAYDATA FROM PAYDATA-REC
  PARENT IS EMPLOYEE.
```

In this example, the previously defined record EMPLOYEE is the parent of child record PAYDATA.

- Using the PARENT clause* You can name a defined record as the parent of another defined record only when, in the source records:
- Repeating group item as child's source*

 - the source (WKLY-PAY in Figure 3-8) of the child record (PAYDATA) is a repeating group item within the group that is the source (EMPLOYEE-REC) of the parent record (EMPLOYEE) or one of the parent's supplements;
 - Two record types as source of parent and child*

 - the source (EMPLOYEE-REC in Figure 3-9) of the parent record (EMPLOYEE) or one of the parent's supplements and source (PAY-REC) of the child (PAYDATA) are two distinct record types (01-level entries) in the same indexed file (EMPFIL);
 - Control break as source*

 - the source (EMPLOYEE-LOC-REC in Figure 3-10) of the parent record (EMPLOYEE) is a control break detected while reading the source of the child (PAYDATA); or
 - Different indexed files as source*

 - the source (PAYDATA-REC in Figure 3-11) of the child record (PAYDATA) is a sequence of records in a different indexed file than the source (EMPLOYEE-REC) of the parent (EMPLOYEE) or any of its supplements; a POINTER clause (3.15) is then used in the child record's defined record definition.

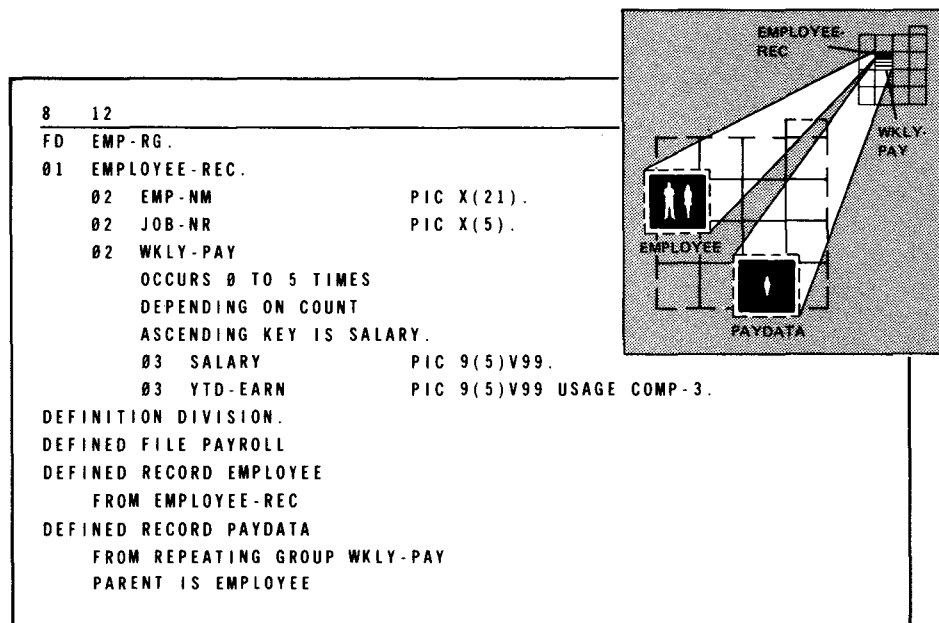


Figure 3-8. Sample Data Definition Using Repeating Group as Source of Child Record

DEFINED RECORD DEFINITION

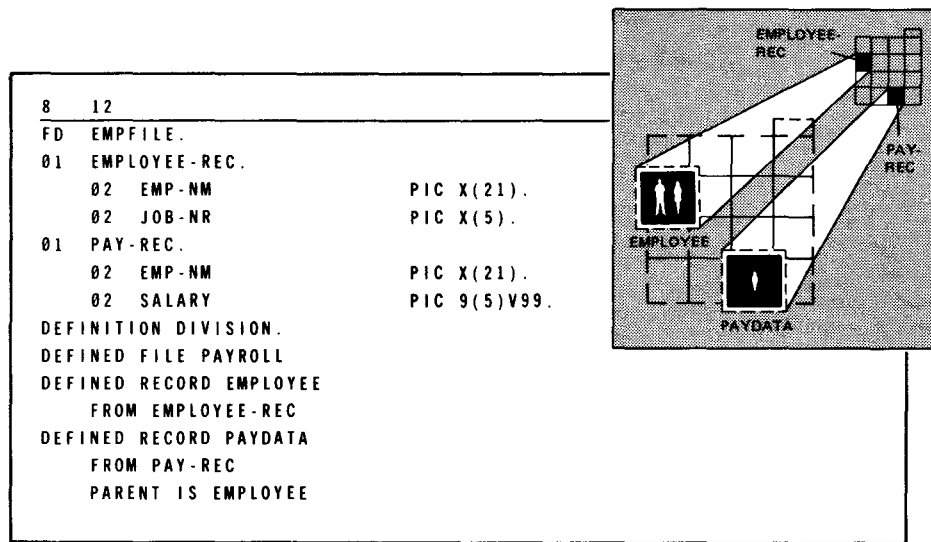


Figure 3-9. Sample Data Definition Using Two Record Types as Sources of Parent and Child Records

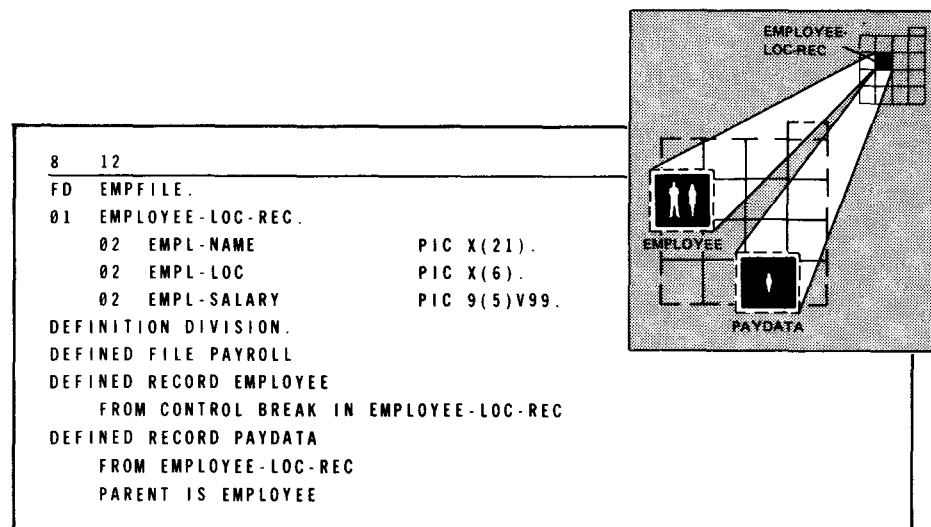


Figure 3-10. Sample Data Definition Using Control Break as Source of Parent Record

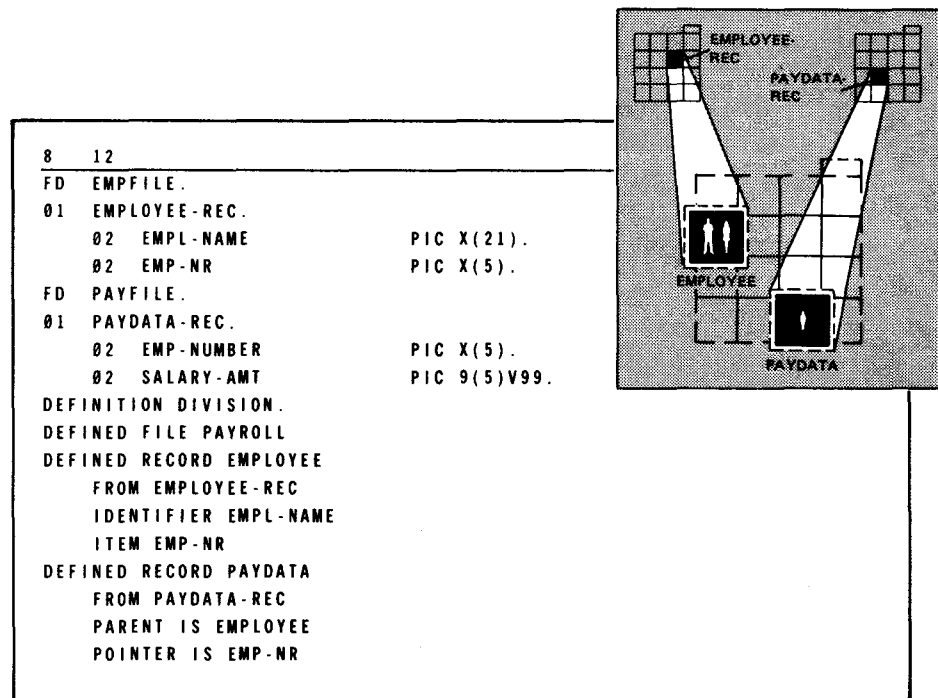


Figure 3-11. Sample Data Definition Using Records in Different Indexed Files as Sources of Parent and Child Records

3.14. DESIGNATING THE PROCESSING ORDER OF FRATERNAL RECORDS (PREFIX CLAUSE)

Function With the PREFIX clause (Figure 3-12), identifier values of fraternal records reflect the order the fraternal records are processed in. It adds to the defined record identifier a character (or characters) not present in any physical record.

The format is:

Format PREFIX IS literal-2

Literal-2 Literal-2 is a constant enclosed by single quotes and added to the identifier of a fraternal-type record.

DEFINED RECORD DEFINITION

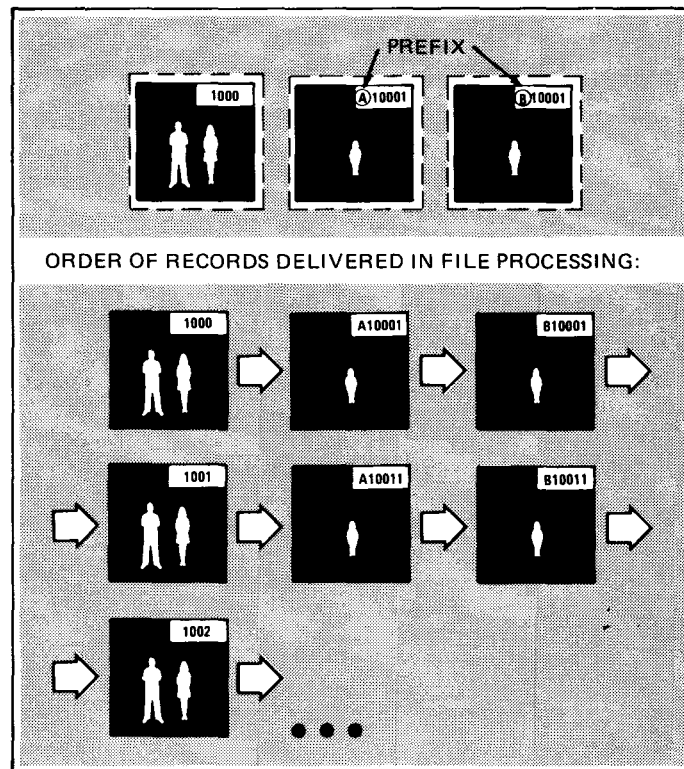


Figure 3-12. PREFIX Clause. The prefix defined in this clause appears in the defined record identifier as shown in the action program or on a terminal screen. It is not part of the physical record's identifier.

Required uses

Include the PREFIX clause, the VALUE clause (3.28), or both, for each defined record that is fraternal to another defined record. You must use the PREFIX clause when value ranges for identifiers of fraternal record types overlap. This occurs when records have sources in different files or in different repeating group items.

Defining successive records

Give the same length prefix values for fraternal record types and put them in ascending order as you define successive records. The prefix you define appears in the identifier of each defined record occurrence. It comes directly before the identifier item in the defined record's first item definition. (See 3.19.)

Example

```

8 12
DEFINITION DIVISION.
DEFINED FILE PAYROLL PASSWORD
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
DEFINED RECORD DEPENDENTS FROM DEPENDENT-REC
    PARENT IS EMPLOYEE
    PREFIX IS 'A'
    IDENTIFIER EMP-NO
DEFINED RECORD PAYDATA FROM PAY-REC
    PARENT IS EMPLOYEE
    PREFIX IS 'B'
    IDENTIFIER EMP-NO

```

Defined file PAYROLL contains three types of records: EMPLOYEE, DEPENDENTS, and PAYDATA. EMPLOYEE is the parent of both DEPENDENTS and PAYDATA, which are fraternal records. Sequential processing of the file delivers all DEPENDENTS records for parent record EMPLOYEE before any PAYDATA records. Prefix 'A' for DEPENDENTS alphabetically comes before prefix 'B' for PAYDATA records; DEPENDENTS records come before PAYDATA records in the defined file. Thus, these prefixes support the requirement for having defined record identifiers in ascending order.

3.15. LOCATING THE SOURCE OF A CHILD DEFINED RECORD (POINTER CLAUSE)

Purpose

In the defined record definition, you use the POINTER clause (Figure 3-13) only for a child defined record. It locates the source of that defined record. Use it when:

- there is a PARENT clause for a defined record; and
- there is a break between the keys of the child's source and the parent's source (the records are in different files or in different locations within the same file).

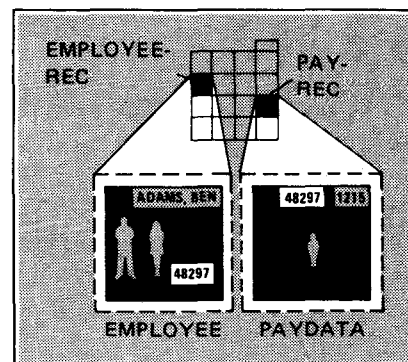


Figure 3-13. POINTER Clause

DEFINED RECORD DEFINITION

Its format is:

Format POINTER IS item-name-1 [, item-name-2]

Item-name Item-name is a term defined in a direct ancestor of this defined record. (Item-names are defined in 3.21 and 3.24.) To retrieve the child defined record's primary part, IMS links the values of item-name-1, item-name-2,... in a character string. Then:

How pointers work

- when the source is an indexed record, characters to the left of the child record's identifier make up the string; or
- when the source is a repeating group item, the string's leftmost characters locate the record occurrence containing the source. When a repeating group item is nested within a larger group item, extra characters are used in the pointer to locate the larger group item by its key.

Example

```

8 12
-----
FD EMPFILE.
01 EMPLOYEE-REC.
   02 EMP-NM                PIC X(21).
   02 EMP-NR                PIC X(5).
FD PAYFILE.
01 PAY-REC.
   02 EMP-NUMBER            PIC X(5).
   02 JOB-NUMBER            PIC X(4).
   02 SALARY-AMT            PIC 9(5)V99.
DEFINITION DIVISION.
DEFINED FILE PAYROLL PASSWORD
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC }
  IDENTIFIER EMP-NM                       } Key is EMP-NM
  ITEM EMP-NR HIDDEN                       }
DEFINED RECORD PAYDATA FROM PAY-REC      }
  PARENT IS EMPLOYEE                       } Key is
  POINTER IS EMP-NR                         } JOB-NUMBER
  IDENTIFIER JOB-NUMBER                     }
  ITEM SALARY-AMT                           }

```

EMPLOYEE is the parent record of PAYDATA. The source of the primary part of EMPLOYEE is in EMPFILE; the source of PAYDATA is in a different indexed file, PAYFILE, ordered by EMP-NUMBER and JOB-NUMBER. The item EMP-NR, already defined in EMPLOYEE, contains the employee number pointing to the pay record needed for each employee. See 3.21 through 3.24 for an explanation of IDENTIFIER and ITEM clauses.

3.16. LOCATING THE SOURCE OF A DEFINED RECORD (FOLLOWS CLAUSE)

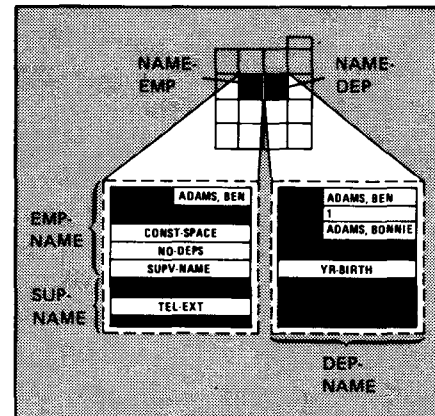
Function

While the POINTER clause connects defined records with sources in different locations, the FOLLOWS clause connects defined records whose sources are two record types that are next to each other in the same indexed file. Use the FOLLOWS clause when the source of this defined record:

- sequentially follows the source of a previous primary part or supplement in this file; but
- there is an intervening defined record or supplement, even if its source is the same record type as the source this record follows.

Restrictions on use

Never use this clause in the first defined record definition for a defined file or when the defined record's source is a repeating group item.



The format is:

Format

```
FOLLOWS { defined-record-name-3 }
         { supplement-name-1   }
```

The source of the current defined record's primary part sequentially follows:

Defined-record-name-3

- the source of a previous defined record when you use defined-record-name-3; or

Supplement-name-1

- the source of a previous supplement when you use supplement-name-1.

DEFINED RECORD DEFINITION*Example*

```

8 12
FD EMPFILE.
01 NAME-EMP.
    02 EMPL-NAME          PIC X(21).
    02 CONST-SPACE       PIC X(2).
    02 NO-DEPS           PIC 9(2).
    02 SUPV-NAME         PIC X(21).
    02 TEL-EXT           PIC 9(4).
01 NAME-DEP.
    02 EMPL-NAME         PIC X(21).
    02 DEP-NO            PIC 9(2).
    02 DEP-NAME          PIC X(21).
    02 YR-BIRTH          PIC 9(4).
DEFINITION DIVISION.
DEFINED FILE EMPLOYEE PASSWORD
DEFINED RECORD EMP-NAME FROM NAME-EMP
    IDENTIFIER EMPL-NAME
    ITEM CONST-SPACE
    ITEM NO-DEPS
    ITEM SUPV-NAME
SUPPLEMENT SUP-NAME FROM NAME-EMP
    POINTER IS SUPV-NAME
    ITEM TEL-EXT
DEFINED RECORD DEP-NAME FROM NAME-DEP
    PARENT IS EMP-NAME
    FOLLOWS EMP-NAME
    IDENTIFIER DEP-NO
    IDENTIFIER DEP-NAME
    ITEM YR-BIRTH

```

} Key is EMPL-NAME

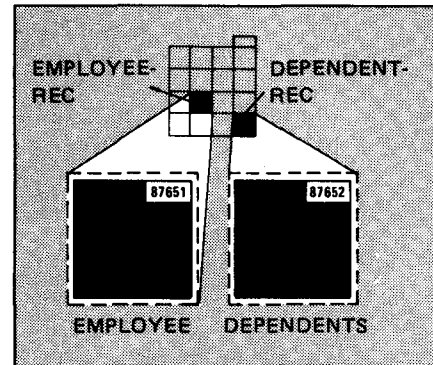
} Key is EMPL-NAME,
DEP-NO, and DEP-NAME

A personnel application has an indexed file (EMPFIL) containing two types of records: employee (NAME-EMP) and dependent (NAME-DEP). In the source file, an employee record is followed by corresponding dependent records. In the defined file (EMPLOYEE), an employee record (for example, ADAMS,BEN) is first followed by a supervisor supplement (VAUGHN,ART), then by dependent records (ADAMS,BONNIE). In the data definition, defined record EMP-NAME names NAME-EMP as its source. Supplement SUP-NAME also names NAME-EMP as its source. Finally, defined record DEP-NAME, defined after SUP-NAME, names NAME-DEP as its source. The FOLLOWS clause tells IMS to read dependent records that follow EMP-NAME (ADAMS,BEN), not SUP-NAME (VAUGHN,ART). See 3.21 and 3.30 for explanations of item and supplement definitions.

3.17. ACCESSING RECORDS ACCORDING TO TYPE (FILL KEY CLAUSE)

Purpose

Indexed records of several types can be interspersed in the source file. The FILL KEY clause allows you to access only those records of a specific type in that file. FILL KEY uses the rightmost characters of an indexed record key to make the key for each record type unique.



The format is:

Format

FILL KEY TO *literal-3*

Literal-3

Literal-3 is the rightmost character or characters of an indexed file's key. Enclose it in single quotes. It can be any character or characters consistent with the PICTURE clause (see Figure 3-2) specified for that identifier.

Using the FILL KEY clause

You only need this clause when:

- the indexed file record key is longer than the combined length of any POINTER and IDENTIFIER items; and
- the remaining characters in the key are not all spaces (hexadecimal 40).

How FILL KEYS work

Literal-3 can be no longer than the part of the key not specified by POINTER or IDENTIFIER items. When creating a search key, IMS fills the remaining character positions with spaces and moves literal-3 into the record key's rightmost character positions.

DEFINED RECORD DEFINITION

Example

```

8 12
FD EMPFILE.
01 EMPLOYEE-REC.
    02 EMP-NR                PIC X(5).
.
.
.
01 DEPENDENT-REC.
    02 EMP-NO                PIC X(5).
.
.
.
DEFINITION DIVISION.
DEFINED FILE PAYROLL PASSWORD
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
    FILL KEY TO '1'
    IDENTIFIER EMP-NR
.
.
.
DEFINED RECORD DEPENDENTS FROM DEPENDENT-REC
    IDENTIFIER EMP-NO
.
.
.

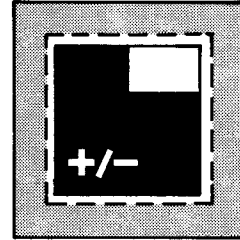
```

Assume EMPLOYEE-REC and DEPENDENT-REC are two types of records in an indexed file (EMPFIL). EMPFILE is the source of defined records EMPLOYEE and DEPENDENTS. Both EMPLOYEE and DEPENDENTS have record keys in the same character positions, 1 through 5. The values of these keys differ only in character position 5. The key to EMPLOYEE-REC is 87651; the key to DEPENDENT-REC is 87652. By defining record EMPLOYEE with the FILL KEY TO '1' clause, you can access EMPLOYEE-REC records by specifying a key of 8765. IDENTIFIER clauses are described in 3.21.

3.18. ALLOWING RECORD ADDITIONS AND DELETIONS (ALLOW ADD AND DELETE CLAUSE)

Purpose

The ALLOW ADD AND DELETE clause permits terminal operators using UNIQUE or your action programs to add or delete defined record occurrences.



The clause is invalid when you use:

- the FROM CONTROL BREAK or FROM REPEATING GROUP format of the FROM clause for a defined record; or
- the FROM REPEATING GROUP format of the FROM clause for any of its supplements.

The format is:


Format

ALLOW { ADD
DELETE
ADD AND DELETE } OF RECORD.

Parameters

ADD  Allows only additions.

DELETE  Allows only deletions.

ADD AND DELETE  Allows additions and deletions.

Adding and deleting records

Without any of these statements in a record definition, you cannot add or delete occurrences of that defined record. For example, when you do not include ALLOW ADD, ALLOW DELETE, or ALLOW ADD AND DELETE in the record definition, IMS rejects as invalid:

- any input of the UNIQUE ADD or DELETE commands; or
- any add or delete function issued by an action program.

Example

```
8 12
DEFINED FILE PAYROLL
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
ALLOW ADD AND DELETE
```

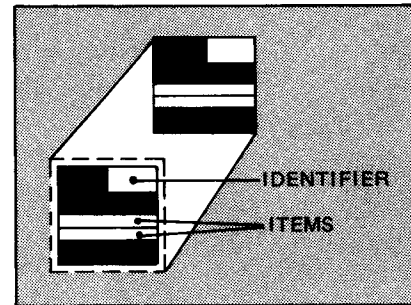
This allows you to add or delete defined record EMPLOYEE in defined file PAYROLL.

ITEM DEFINITION

3.19. ITEM DEFINITION

Function

A defined record consists of an identifier and other items taken from the same record and other records. Item definitions describe these items. IDENTIFIER statements (3.20 through 3.22) name identifier items; ITEM statements (3.23 through 3.28) name other items in the defined record. Figure 3-14 shows the item definition format.

*Format*

```

{ IDENTIFIER [ item-name-1 FROM ] data-name-1 }
{ ITEM      [ item-name-2 FROM ] data-name-2 }
[ HIDDEN ]
[ MUST ADD ]
[ ALLOW CHANGE ]
[ VALUE IS ] literal-1 [ THROUGH ] literal-2 ]
[ VALUES ARE ] [ THRU ]
[ , literal-3 [ THROUGH ] literal-4 ] ... ]
[ ALSO [ item-alias-1 FROM ] item-name-3 ]
[ , [ item-alias-2 FROM ] item-name-4 ] ... ]

```

Figure 3-14. Item Definition Format

Rule

You must write a separate item definition for each identifier and item name in the defined record. Each defined record can have up to 78 IDENTIFIER and ITEM statements.

UNIQUE column headers

When using UNIQUE, carefully consider the size and meaning of item names, because they are displayed as column headers in all UNIQUE command response output. Also, keep in mind that UNIQUE uses one extra space for signed number items, and another extra space for tab stop control characters. In the terminal display, UNIQUE inserts two spaces between column headers or data items, whichever are longer.

Identifier items

Defined record identifiers locate the data in your source files. Identifiers can come from any field that is part of the record key. Because they are derived from key fields, identifier items can only come from records in:

- indexed files; or
- a data base subschema.

Data definitions using a subschema as source are discussed in the IMS/DMS interface user guide, UP-8748 (current version).

Simple defined files

Figure 3-15 shows how a defined record identifier is derived from one record in a simple defined file. It also shows how other items in the defined record are derived from fields in the same source record. Figure 3-16 shows a terminal display of item names as column headers plus data from the defined record.

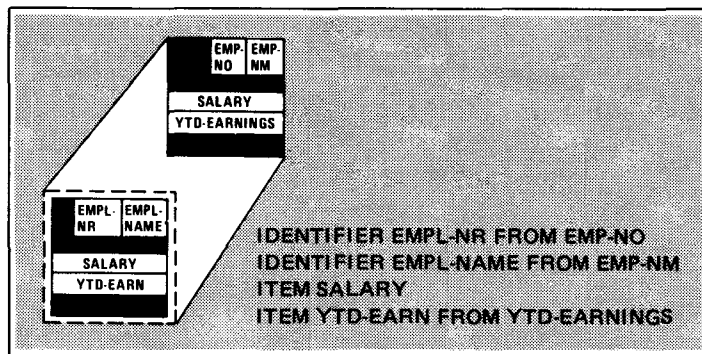


Figure 3-15. Defined Record Identifier in a Simple Defined File

2 SPACES

| * EMPL-NR | EMPL-NAME | SALARY | YTD-EARN |
|-----------|----------------|--------|----------|
| . 60155 | HUNTER, SHARON | 205.00 | 5,877.00 |

Figure 3-16. Terminal Display of Column Headers and Data Items in a Simple Defined File

ITEM DEFINITION

**Simple vs.
hierarchical
defined files**

While a simple defined file contains only one type of defined record, a hierarchical defined file contains two or more types of defined records that have a parent-child relationship (2.4).

Parent-child identifiers

The entire parent record identifier is carried down to the beginning of its child's record identifier. Remaining items in child identifiers distinguish between child records. These unique portions of child identifiers may come from different files than parent identifiers.

Hierarchical defined files

Figure 3-17 compares parent and child record identifiers in a hierarchical defined file. It also shows how their identifiers are derived from records in two different indexed files. Figure 3-18 shows the terminal display of parent-child defined record identifiers. For more examples of parent-child records, see 4.2.

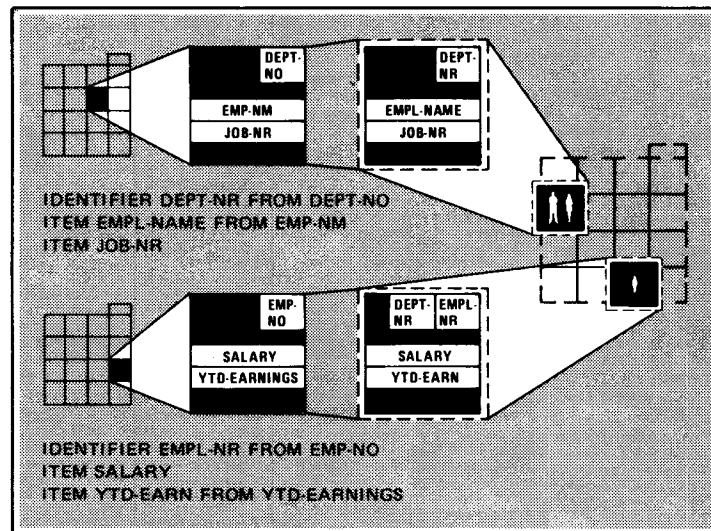
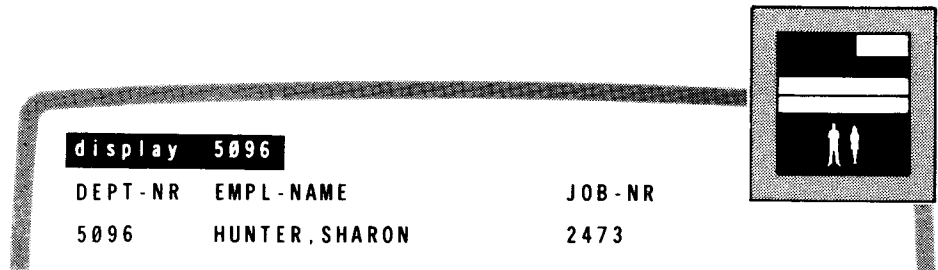
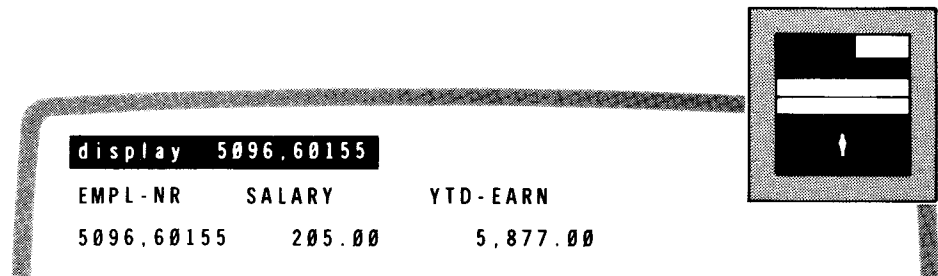


Figure 3-17. Parent-Child Defined Record Identifiers



a. Parent record displayed



b. Child record displayed

Figure 3-18. Terminal Displays of Column Headers and Data Items for Parent and Child Records

3.20. DEFINING AN IDENTIFIER (IDENTIFIER STATEMENT)

Figure 3-19 shows the format for an IDENTIFIER statement.

Format

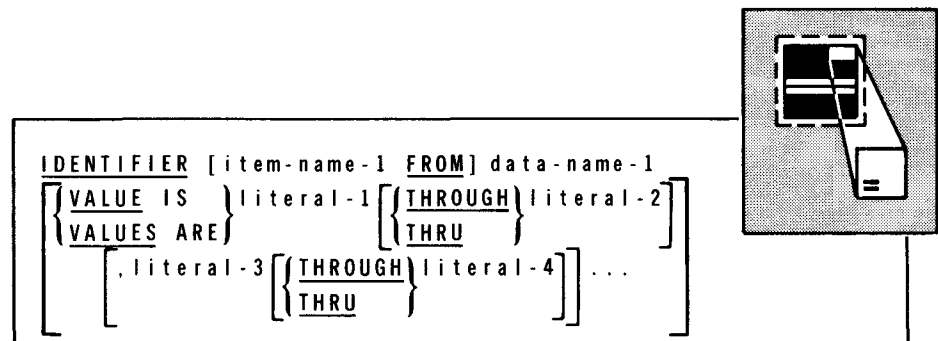


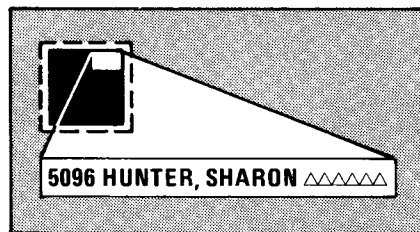
Figure 3-19. IDENTIFIER Statement Format

ITEM DEFINITION

3.21. NAMING THE IDENTIFIER (IDENTIFIER CLAUSE)

Function

The IDENTIFIER clause begins an IDENTIFIER statement and names the defined record identifier item. You must specify identifier items before any other defined record items. The format is:

*Format*

IDENTIFIER [*item-name-1* FROM] *data-name-1*

Item-name-1

Item-name-1 is a 1- to 30-alphanumeric character name, unique within the defined file definition. UNIQUE uses it as a terminal column header. You can omit item-name-1 when it is identical to data-name-1.

Data-name-1

Data-name-1 is a data division data name that is part of the source of this defined record's primary part. When the source is:

- A record described with the FROM or FROM CONTROL BREAK format of the FROM clause, data-name-1 must be part of that record's key.
- A repeating group item, described with the FROM REPEATING GROUP format of the FROM clause, data-name-1 must be part of that group item's key.

Example

```

8 12
FD EMPFILE.
01 EMPLOYEE-REC.
   02 DEPT-NR           PIC X(4).
   02 EMPL-NAME        PIC X(21).
   02 EMPL-NR          PIC X(5).
DEFINITION DIVISION.
DEFINED FILE PAYROLL
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC }
  IDENTIFIER DEPT-NR                       } Key is DEPT-NR
  IDENTIFIER EMP-NM FROM EMPL-NAME         } and EMPL-NAME
  ITEM EMPL-NR

```

Multiple identifiers

When you use more than one IDENTIFIER clause, define items in major-to-minor order. In this example, DEPT-NR is the major identifier; EMP-NM is the minor identifier. The value of the record key in this defined record's source record is:

5096HUNTER, SHARON△△△△△△△△

Identifier segments

The sequence of items defined by the IDENTIFIER clause appears at the terminal as a string of identifier segments separated by commas:

```
5096 , HUNTER , SHARON
```

UNIQUE display of multiple identifiers

When you use multiple IDENTIFIER clauses, UNIQUE identifies the entire identifier string by the item name in the final IDENTIFIER clause. For example, if you specify both identifier items DEPT-NR and EMP-NM, both identifier names appear as terminal headers for the string. If you specify the minor identifier only, the terminal operator sees:

- the minor identifier and its header for the DISPLAY command; and
- the minor identifier and both the major and minor identifiers' headers for the LIST command.

Sample displays

Figure 3-20 shows sample terminal displays of multiple identifier items.

```
display 5096 hunter,sharon
DEPT - NR , EMP - NM      EMPL - NR
5096 , HUNTER , SHARON   60155
```

a. Specifying both major and minor identifier items

```
display hunter,sharon
EMP - NM      EMPL - NR
HUNTER , SHARON 60155
```

b. Specifying the minor identifier for the DISPLAY command

```
list emp-nm for 5096
5096
. DEPT - NR , EMP - NM
. HUNTER , SHARON
. HYATT , BARBARA
. JANSSEN , ALEX
```

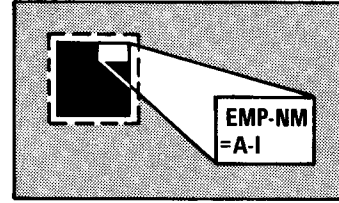
c. Specifying the minor identifier for the LIST command

Figure 3-20. Terminal Display of Column Headers and Data Items for Multiple Identifiers

ITEM DEFINITION

**3.22. SPECIFYING A VALUE RANGE FOR THE IDENTIFIER
(VALUE CLAUSE)***Purpose*

You can use the VALUE clause to specify the valid value ranges an identifier can have when you are adding a record. Before allowing you to add the record, IMS checks its identifier's validity and makes sure that its value lies within the specified ranges.

*Effect of omitting VALUE clause*

When you omit the VALUE clause, IMS accepts any value consistent with the PICTURE and USAGE clauses specified for this item's source. (See Figure 3-2.)

The format is:

Format

$$\left\{ \begin{array}{l} \text{VALUE IS} \\ \text{VALUES ARE} \end{array} \right\} \text{literal-1} \left[\begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right] \text{literal-2} \left[\begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right] \text{literal-3} \left[\begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right] \text{literal-4} \dots$$
Literal-1, literal-2, ...

Literal-1, literal-2, ... specify the values or value ranges allowed for an identifier when a record is being added. You must put the values for literal-1, literal-2, etc, in ascending order and make their lengths equal to:

- each other; and
- to the item named by data-name-1 or data-name-2 in the ITEM or IDENTIFIER statement.

Enclosing in quotes

Alphanumeric literals have to be enclosed in quotes; numeric literals do not.

Using the VALUE clause

You must use the VALUE clause in an IDENTIFIER statement when:

- you need to distinguish between indexed record occurrences that contribute to the defined record and successive occurrences of the same indexed record that do not contribute to the defined file; or
- fraternal record types that have the same source (however, their value ranges cannot overlap).

Use with fraternal records

When you use the VALUE clause for identifier items in fraternal records that:

- come from different sources, and
- have identifier value ranges that overlap,

you must also include the PREFIX clause (3.14).

Use with file segmentation

When you use the VALUE clause for an IDENTIFIER item, you cannot access records with keys outside the specified range. Thus, you could use the VALUE clause for file segmentation. For example, you could specify value ranges for IDENTIFIER items as A through I, J through R, and S through Z, to process segments of your payroll file in stages.

Example

```

8 12
-----
DEFINED RECORD PAYDATA FROM PAY-REC
  ALLOW ADD AND DELETE
  IDENTIFIER EMP-NM VALUE IS 'A' THROUGH 'I'
    
```

In this example, you can add a pay record, but, unless the record identifier's value falls between A and I, IMS rejects the update and returns an invalid request indicator (003) in the program status code.

3.23. DEFINING OTHER ITEMS IN THE DEFINED RECORD (ITEM STATEMENT)

Figure 3-21 shows the format for an ITEM statement.

Format

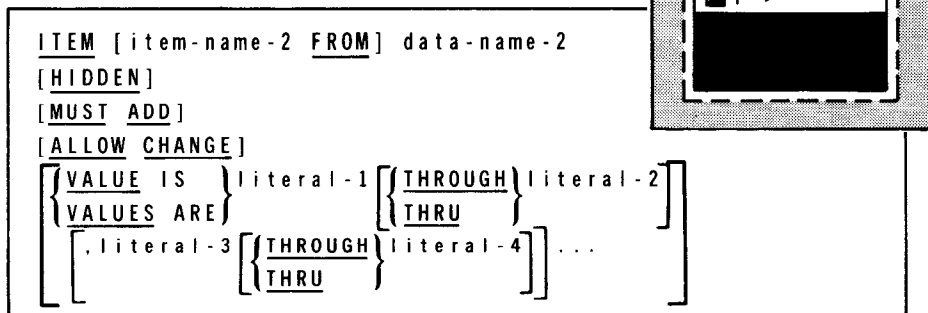
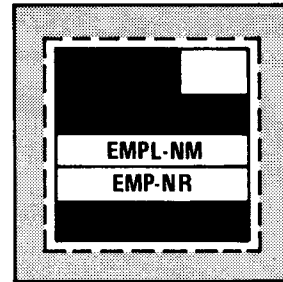


Figure 3-21. ITEM Statement Format

ITEM DEFINITION

3.24. NAMING THE ITEM (ITEM CLAUSE)*Function*

The ITEM clause begins an ITEM statement and names an item in the defined record. Its format is:

*Format*

ITEM [item-name-2 FROM] data-name-2

Item-name-2

Item-name-2 is a 1- to 30-character name, unique within the defined file definition. When you use UNIQUE, it appears as a terminal column header. You can omit item-name-2 when it is identical to data-name-2.

Data-name-2

Data-name-2 is a data division data name that is part of the source of the defined record's primary part. Never qualify data-name-2; source name qualification is implied. Data-name-2 must be:

- an elementary item; or
- a group item that contains only alphabetic, alphanumeric, or numeric items specifying USAGE IS DISPLAY.

Coding rule

The data of the item named in the ITEM clause should not exceed:

- 72 characters; or
- 2 characters less than line length when UNIQUE displays it on a terminal containing less than 74 characters per line.

Example

```

8 12
DATA DIVISION.
FILE SECTION.
FD EMPFILE.
  01 EMPLOYEE-REC.
     02 DEPT-NR                PIC X(4).
     02 EMP-NM                 PIC X(21).
     02 EMP-NR                 PIC X(5).
DEFINITION DIVISION.
DEFINED FILE PAYROLL
DEFINED RECORD EMPLOYEE
  FROM EMPLOYEE-REC
  IDENTIFIER DEPT-NR
  ITEM EMPL-NAME FROM EMP-NM
  ITEM EMP-NR

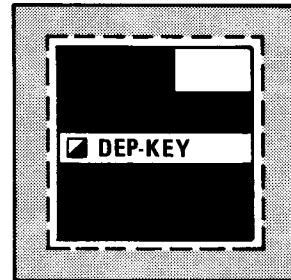
```

Defined record EMPLOYEE includes items EMPL-NAME and EMP-NR from EMPLOYEE-REC. Item EMP-NM is renamed EMPL-NAME; item EMP-NR retains the same name.

3.25. PREVENTING ITEM DISPLAY (HIDDEN OPTION)

Purpose

When using UNIQUE, you can prevent the terminal display of a data item defined by an ITEM statement. The HIDDEN option allows a subsequent POINTER clause to refer to an item without having that item displayed. The format is:

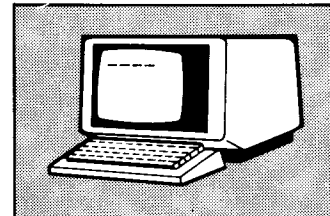


Format

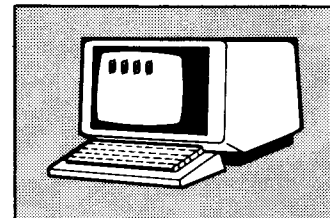
HIDDEN

Using the HIDDEN option

When a UNIQUE terminal operator adds a defined record containing an item definition that specifies the HIDDEN option, he sees:



- spaces where an alphanumeric item would otherwise appear; and
- zeros (in the proper data format) where a numeric item would appear.



Validating numeric fields

You can also use the HIDDEN option to place the correct format in a numeric field in a record that you add. Normally, IMS inserts binary zeros in the missing fields not included in a defined record. However, you can prevent this by including these fields in the defined record with ITEM clauses and then restricting their use with the HIDDEN option.

Action programs and identifiers

IMS ignores the word HIDDEN when:

- you access a defined record with your own action programs; or
- you use it in an IDENTIFIER statement.

ITEM DEFINITION*Example*

```

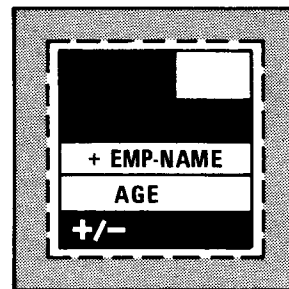
8 12
-----
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
  IDENTIFIER EMP-NR
  ITEM DEP-KEY HIDDEN
DEFINED RECORD DEPENDENTS FROM DEPENDENT-REC
  PARENT IS EMPLOYEE
  POINTER IS DEP-KEY

```

In this example, assume each employee record gives a pointer (DEP-KEY) that is used to locate the set of dependent records that are its child records. You can use the HIDDEN option when you do not want DEP-KEY data displayed at a UNIQUE terminal.

3.26. SPECIFYING A REQUIRED ITEM (MUST ADD OPTION)*Purpose*

You can use the MUST ADD option to specify that a certain record item must be present and contain a valid value before you add a record to the defined file. Use it only in an ITEM statement; identifier items are always present when you add a record. The format is:

*Format*

MUST ADD

Using the MUST ADD option

To be valid, a numeric item must be nonzero, and an alphanumeric item must contain other than all spaces.

ALLOW ADD clause

This option works only when you specify the ALLOW ADD clause (3.18) in your defined record definition.

Example

```

8 12
-----
DEFINED FILE PAYROLL
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
  ALLOW ADD AND DELETE
  IDENTIFIER EMP-NR
  ITEM EMP-NAME MUST ADD
  ITEM AGE

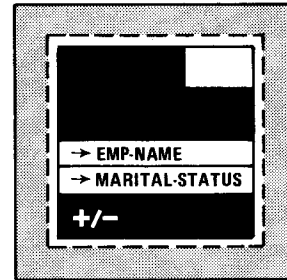
```

Before you add an EMPLOYEE record to defined file PAYROLL, item EMP-NAME must contain a valid value. In this application, you use the MUST ADD option for EMP-NAME because it is an important part of the employee record. You don't specify the MUST ADD option for item AGE because, for this particular application, that item is not of major importance to the record.

3.27. ALLOWING CHANGES TO THE ITEM (ALLOW CHANGE OPTION)

Purpose

Terminal operators can only change the current item's value when you specify the ALLOW CHANGE option. The format is:

**Format**

ALLOW CHANGE

Using the ALLOW CHANGE option

This option works only in an ITEM statement, because identifier items cannot be changed.

Restrictions on use

Do not specify ALLOW CHANGE in ITEM statements for two items in a defined record when:

- The source of one item overlaps the source of the other item (either item is a group item that contains the other). Otherwise, when item values are moved to a new or updated source record, the second item moved covers up the first.
- Both items have the same source field on disk. Otherwise, when you try to update the two items to new values, you don't know what value you will get on disk.

Example

```

8 12
DEFINED RECORD EMPLOYEE
  ALLOW ADD AND DELETE
  IDENTIFIER EMP-NR
  ITEM EMP-NAME ALLOW CHANGE
  ITEM MARITAL-STATUS ALLOW CHANGE

```

This example specifies that you can change items EMP-NAME and MARITAL-STATUS in defined record EMPLOYEE. You cannot change EMP-NR because it's an identifier item.

Changes to items

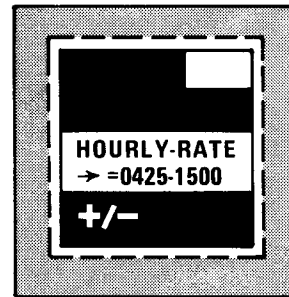
Without ALLOW CHANGE, IMS won't carry out any requested item value changes to records on disk. When an action program attempts to update a record containing the changed item value, IMS returns control to the action program with an invalid request indicator (003) in the program status code.

ITEM DEFINITION

3.28. SPECIFYING A VALUE RANGE FOR THE ITEM (VALUE CLAUSE)

Purpose

You can use the VALUE clause to specify the valid value ranges an item can have when you are adding or changing it. Before allowing you to update (ADD, CHANGE, PUT, or INSERT) an item, IMS checks its validity and makes sure that its value lies within the specified ranges.

*Effect of omitting VALUE clause*

When you omit the VALUE clause, IMS accepts any value consistent with the PICTURE and USAGE clauses specified for this item's source. (See Figure 3-2.)

The format is:

Format

$$\left\{ \begin{array}{l} \text{VALUE IS} \\ \text{VALUES ARE} \end{array} \right\} \text{literal-1} \left[\begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right] \text{literal-2} \left[\begin{array}{l} \text{THROUGH} \\ \text{THRU} \end{array} \right] \text{literal-4} \dots$$
Literal-1, literal-2, ...

Literal-1, literal-2, ... specify the values or value ranges allowed for an item being added or changed. You must put the values for literal-1, literal-2, etc, in ascending order and make their lengths equal to:

- each other; and
- the item named by data-name-1 or data-name-2 in the ITEM or IDENTIFIER statement.

Enclosing in quotes

Alphanumeric literals have to be enclosed in quotes; numeric literals do not.

Example

```

8 12
DEFINED RECORD PAYDATA FROM PAY-REC
  ALLOW ADD AND DELETE
  IDENTIFIER EMP-NR
  ITEM HOURLY-RATE ALLOW CHANGE VALUE IS 0425 THROUGH 1500

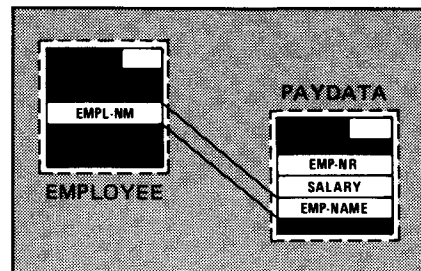
```

In this example, you can change the item HOURLY-RATE, but, unless the new values fall between 425 and 1500, IMS rejects the update.

3.29. INCLUDING PREVIOUSLY DEFINED ITEMS IN THE DEFINED RECORD (ALSO CLAUSE)

Purpose

When you use the ALSO clause, the current defined record can include items described in its direct ancestors' definitions. Without it, you can only include these items in the defined record by using a supplement definition (2.30).



Coding rule

This clause follows a defined record's item definitions.

Its format is:

Format

```
ALSO [item-alias-1 FROM]item-name-3
     [, [item-alias-2 FROM]item-name-4]...
```

Item-alias, item-name

For item-alias, use a 1- to 30-character name, unique for an item in the defined file. For item-name, use an item defined in a direct ancestor's defined record definition.

Example

```
8 12
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
  IDENTIFIER JOB-NR
  ITEM EMPL-NM
DEFINED RECORD PAYDATA FROM PAY-REC
  PARENT IS EMPLOYEE
  IDENTIFIER DEPT-NR
  ITEM EMP-NR
  ITEM SALARY
  ALSO EMP-NAME FROM EMPL-NM
```

The ALSO clause includes item EMP-NAME in record PAYDATA. EMP-NAME was previously defined as item EMPL-NM in ancestor record EMPLOYEE.

SUPPLEMENT DEFINITION**3.30. SUPPLEMENT DEFINITION****Purpose**

Your defined record already contains identifier and other items from the primary part's source record. By using defined record supplements, you can include in your defined record additional items from either indexed or nonindexed files. These additional items can come from:

Sources of additional data items

- the same source record;
- a different source record; or
- a repeating group in different source record.

Items can also come from a data base subschema, which is discussed in the IMS/DMS interface user guide, UP-8748 (current version).

Rule

You must write separate supplement definitions for items that come from different sources and are added to a defined record.

Using multiple defined files

Supplements also help you to use multiple defined files to access data in different ways. See the IMS concepts and facilities manual, UP-9205 (current version) for a more detailed explanation of interrelated files.

Figure 3-22 shows the format of the supplement definition and the sequence its clauses must follow.

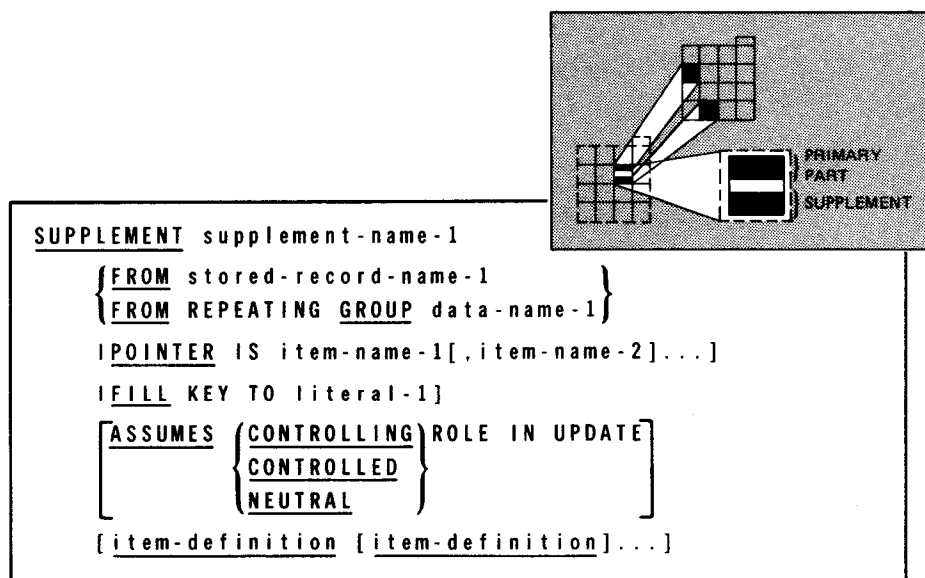
Format

Figure 3-22. Supplement Definition Format

Deriving a supplement item

Figure 3-23 shows how an item is derived from a different source record and included in a defined record.

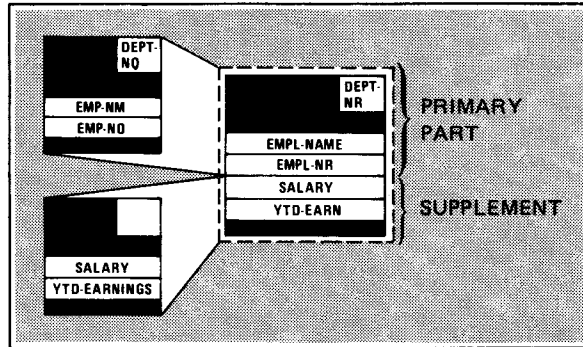


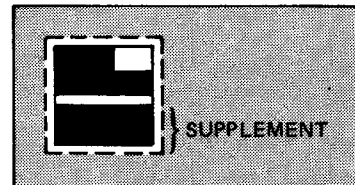
Figure 3-23. Deriving an Item from Another Source Record

You cannot begin a supplement item definition with an IDENTIFIER statement; it can only contain ITEM statements.

3.31. NAMING THE SUPPLEMENT (SUPPLEMENT CLAUSE)

Function

The SUPPLEMENT clause begins a supplement definition and names the supplement. Starting in column 8, the format is:



Format

SUPPLEMENT supplement-name-1

Supplement-name-1

Supplement-name-1 is a 1- to 30-character name, unique within the data definition.

Example

```
8 12
SUPPLEMENT DEPENDENT
```

This example identifies a supplement, DEPENDENT.

SUPPLEMENT DEFINITION**3.32. IDENTIFYING THE SOURCE OF THE SUPPLEMENT
(FROM CLAUSE)***Function and formats*

Put the FROM clause directly after the SUPPLEMENT clause. It specifies the source of the supplement and has two formats, shown in Figure 3-24.

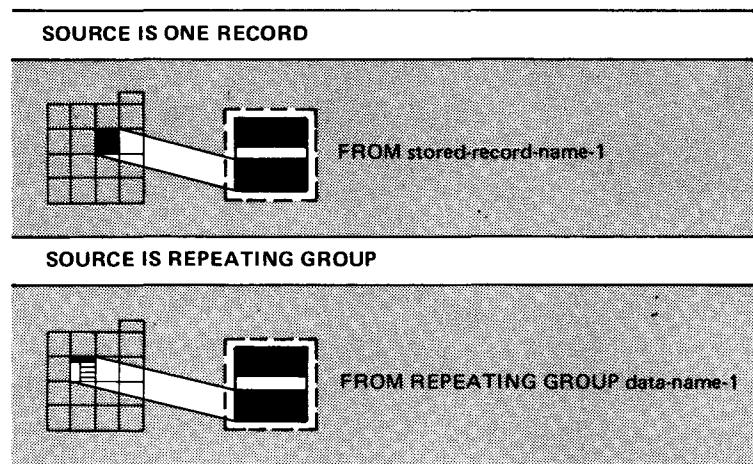


Figure 3-24. Describing the Supplement's Source Using FROM Clause Format

Specifying One Record as the Source

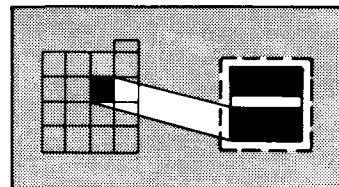
Use the format

Format

FROM stored-record-name-1

Purpose

when the source of the supplement is one record that is the same as or different than the source of the primary part.

*Stored-record-name-1*

Stored-record-name-1 is an O1-level record description in the data division (3.4). This supplement can include a data item from this source record when:

- the data item meets length and usage constraints (3.24); and
- within stored-record-name-1, the data item precedes any item defined with an OCCURS clause (see Figure 3-2).

Example

```

8 12
DATA DIVISION.
FILE SECTION.
FD EMP-FILE.
01 EMPLOYEE-REC.
   02 EMP-NM PIC X(21).
   .
   .
01 DEPENDENT-REC.
   02 DEP-NM PIC X(21).
   .
   .
DEFINITION DIVISION.
DEFINED FILE EMPLOYEES
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
   .
   .
SUPPLEMENT DEPENDENT FROM DEPENDENT-REC
   .
   .

```

DEPENDENT-REC, an 01-level record description, supplies the contents of supplement DEPENDENT.

Specifying a Repeating Group as the Source

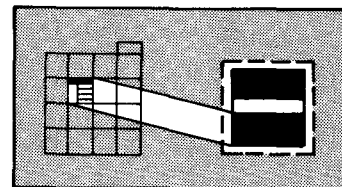
Use the format

Format

FROM REPEATING GROUP data-name-1

Purpose

when the source of the supplement is a group item appearing in the data division with an OCCURS clause. The supplement's source can come from the same or a different file than the source of the primary part.



SUPPLEMENT DEFINITION

Data-name-1

Data-name-1 is a data name defined in the data division with both OCCURS and KEY clauses. You can include any data-name-1 item in this supplement when:

- the item meets length and usage constraints (3.24); and
- within data-name-1, it precedes any item (other than data-name-1 itself) defined with an OCCURS clause.

When data-name-1 is contained within one or two larger group items that are also described with OCCURS clauses, those descriptions must include the KEY clause.

Restrictions on adding records

Do not use this format when you want to add records with a UNIQUE ADD command, INSERT function, or ADD specification in your action program. Adding a record produces binary zeros as the value of data-name-1, so it cannot contain a unique key.

Example

```

8 12
DATA DIVISION.
FILE SECTION.
FD EMP-FILE.
01 EMPLOYEE-REC.
   02 EMP-NR          PIC X(5).
   02 EMP-NM          PIC X(21).
   02 DEPENDENTS OCCURS 10 TIMES
      ASCENDING KEY IS DEP-NAME.
       03 DEP-NAME    PIC X(21).
DEFINITION DIVISION.
DEFINED FILE INSURANCE
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
  IDENTIFIER EMP-NR
  ITEM EMP-NM
SUPPLEMENT DEPENDENT FROM REPEATING GROUP DEPENDENTS

```

Repeating group DEPENDENTS, an 02-level entry in record EMPLOYEE-REC, supplies the contents of supplement DEPENDENT. DEPENDENTS appears in the data division file section with both OCCURS and KEY clauses.

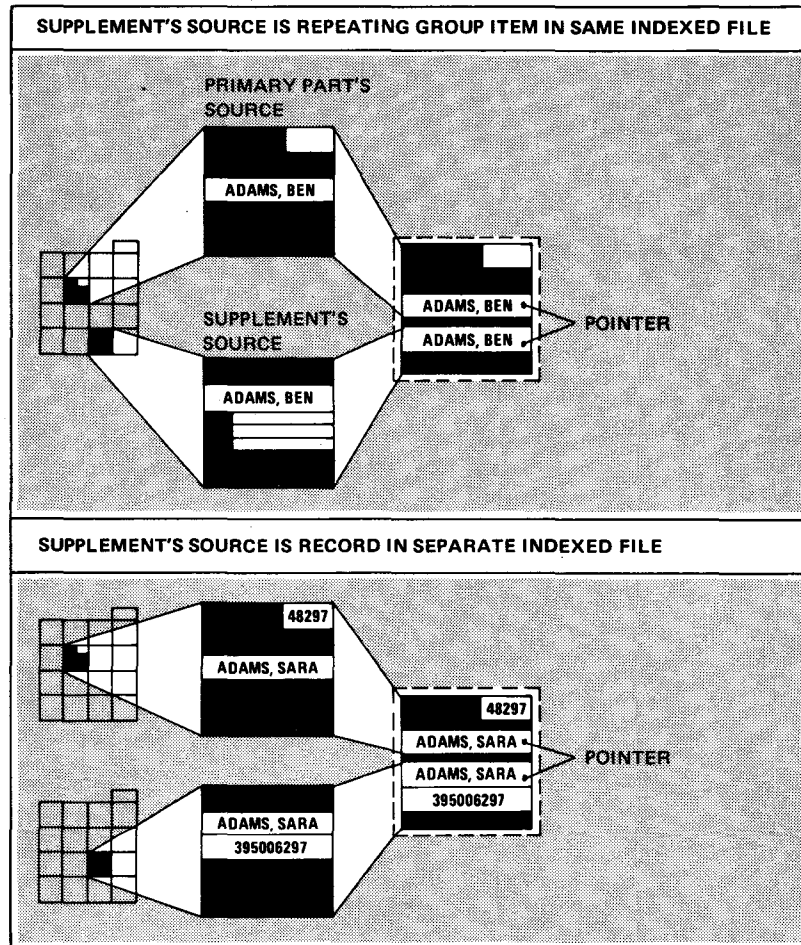
3.33. LOCATING THE SOURCE OF THE SUPPLEMENT (POINTER CLAUSE)

Purpose

When the supplement's source is:

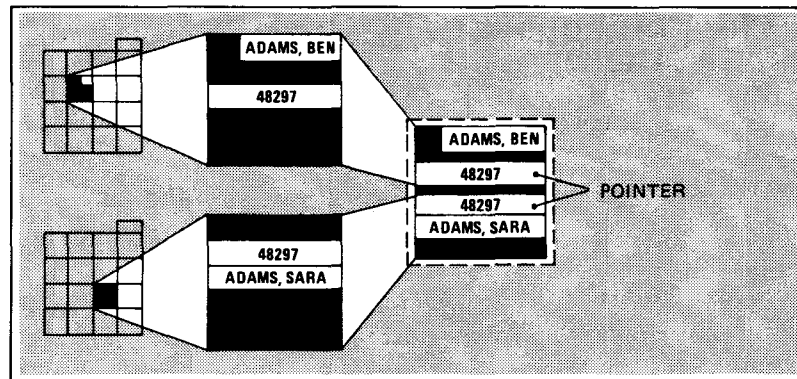
- a repeating group item in the same indexed file as the primary part's source, or
- a record in a different indexed or nonindexed file than the primary part's source,

you must use the POINTER clause (Figure 3-25) to name the items whose values locate a specific occurrence of this supplement's source record.



a. Building an indexed record key

Figure 3-25. POINTER Clause for Supplements (Part 1 of 2)

SUPPLEMENT DEFINITION

b. Building a relative record number

Figure 3-25. POINTER Clause for Supplements (Part 2 of 2)

The format is:

Format `POINTER IS item-name-1 [, item-name-2] ...`

Item-name Item-name-1, item-name-2, ... are items previously defined in the definition of:

- the current defined record; or
- a direct ancestor of the current defined record.

Forming pointers The POINTER is a character string formed by linking the values of item-name-1, item-name-2, ... from left to right.

Separate file or repeating group as source When this supplement's source is in a separate indexed file or is a repeating group item in the same indexed file, IMS builds a reference key by:

- matching the record key in the supplement's source with a field in the primary part's source;
- left-justifying the reference key (pointer) and filling it to the right with spaces (hexadecimal 40); and
- making the pointer's rightmost characters equal to literal-1 in the FILL KEY clause, if specified.

Nonindexed file as source

When this supplement's source is a record or repeating group item in a nonindexed file, IMS builds a file relative record number by:

- matching the relative record number from the supplement's source with a field in the primary part's source; and
- right-justifying the relative record number (pointer) and filling it to the left with binary zeros.

Repeating group requirements

When the supplement's source is a repeating group in either an indexed or nonindexed file, the record keys of the repeating group's records must differ only in the rightmost character positions, as specified by literal-1 of the FILL KEY clause (3.34). The remaining characters in the record keys are identical for all records in the group and are determined by the key of the first source in the group.

Example 1

```

8 12
-----
FD EMPFILE.
01 EMP-REC.
   02 EMP-NO          PIC X(5).
   02 NAME-DEP       PIC X(21).
FD DEPFIL.
01 DEP-REC.
   02 DEPS OCCURS 5 TIMES
      ASCENDING KEY IS DEP-NAME.
   03 DEP-NAME       PIC X(21).
   03 D-SSN          PIC X(9).
DEFINITION DIVISION.
DEFINED FILE EMPLOYEES
DEFINED RECORD EMPLOYEE FROM EMP-REC
  IDENTIFIER EMP-NO
  ITEM NAME-DEP
SUPPLEMENT DEPENDENT FROM REPEATING GROUP DEPS
  POINTER IS NAME-DEP
  ITEM D-SSN

```

This example involves sources in two indexed files. Defined record EMPLOYEE's data comes from EMP-REC in indexed file EMPFILE. Supplement DEPENDENT's data comes from DEPFIL's repeating group DEPS, whose key equals DEP-NAME. Item NAME-DEP contains a record key pointing to the DEPFIL record that holds the dependent data for each employee record.

SUPPLEMENT DEFINITION*Example 2*

```

8 12
FD EMPFILE
01 EMP-REC.
   02 EMP-NM          PIC X(21).
   02 EMP-NO          PIC X(5).
FD DEPFIL.
01 DEP-REC.
   02 REC-NO          PIC X(5).
   02 DEP-NM          PIC X(21).
DEFINITION DIVISION.
DEFINED FILE EMPLS
DEFINED RECORD EMPLOYEE FROM EMP-REC
  IDENTIFIER EMP-NM
  ITEM EMP-NO
SUPPLEMENT DEPENDENT FROM DEP-REC
  POINTER IS EMP-NO
  ITEM DEP-NM

```

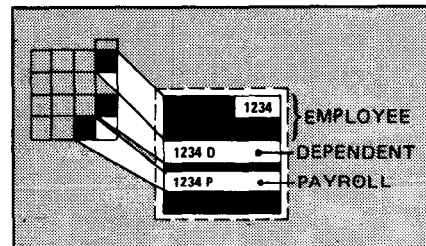
This example involves sources in indexed and nonindexed files. Defined record EMPLOYEE's data comes from EMP-REC in indexed file EMPFILE. Supplement DEPENDENT's data comes from DEP-REC nonindexed file DEPFIL. Item EMP-NO contains a file relative record number pointing to the DEPFIL record that holds the dependent data for each employee record.

*Interrelated
defined files*

The POINTER clause can also have the same effect as an ITEM clause when it is used for supplements in interrelated defined files. See 4.5 for an example.

3.34. ACCESSING RECORDS ACCORDING TO TYPE (FILL KEY CLAUSE)*Purpose*

When several types of indexed records are interspersed in the source file, you can use the FILL KEY clause to distinguish between record types. This clause uses the rightmost characters of an indexed file's record key to make the key for each record type unique.



Using the FILL KEY clause






You must use the FILL KEY clause for records in the same source file when:

- there is no POINTER clause; or
- the POINTER clause does not specify all of the characters of a record key, and the remaining right-hand characters must have a value other than spaces (hexadecimal 40).

POINTER and FILL KEY clauses

Table 3-2 summarizes the uses of FILL KEY and POINTER clauses.

Table 3-2. Using POINTER and FILL KEY Clauses

| Sources of Primary Part and Supplement Are: | POINTER Clause | FILL KEY Clause |
|--|--|--|
| SAME RECORD  | Not required | Not applicable |
| DIFFERENT RECORDS IN SAME FILE  | Not required, but can be used | Required |
| REPEATING GROUP IN SAME FILE  | Required for first record in sequence | Not applicable |
| RECORDS IN DIFFERENT INDEXED FILES  | Required for first supplement named from that file | Required for multiple supplements from same file |
| RECORD IN DIFFERENT INDEXED AND NONINDEXED FILES  | Required | Not applicable |

SUPPLEMENT DEFINITION

The format is:

Format FILL KEY TO literal-1

Literal-1
How FILL KEYS work

Literal-1 becomes the rightmost character or characters of the record key and must be enclosed in single quotes. It can be no longer than the part of the key not specified by POINTER and IDENTIFIER items. When there is no POINTER clause, the value of literal-1 must be:

- greater than spaces (hexadecimal 40); and
- greater than literal-1 of any FILL KEY clause in the directly preceding supplement definition, because each indexed record's key must be greater than the key of the record directly preceding it in the file.

Example 1

```

8  12
FD  EMPFILE.
01  EMP-REC.
     02  EMP-NO           PIC X(5).
01  DEP-REC.
     02  EMP-NO           PIC X(5).
01  PAY-REC.
     02  EMP-NO           PIC X(5).
DEFINITION DIVISION.
DEFINED FILE EMPLS
DEFINED RECORD EMPLOYEE FROM EMP-REC
    IDENTIFIER EMP-NO
SUPPLEMENT DEPENDENT FROM DEP-REC
    FILL KEY TO 'D'
SUPPLEMENT PAYROLL FROM PAY-REC
    FILL KEY TO 'P'

```

In this example, indexed file EMPFILE includes employee, dependent, and payroll records. EMP-REC record keys are emp-no,Δ; DEP-REC record keys are emp-no,D; PAY-REC record keys are emp-no,P. Defined record EMPLOYEE names EMP-REC as its source, but, by specifying FILL KEY TO 'D' and FILL KEY TO 'P', you need no pointers to the sources of DEPENDENT or PAYROLL.

Example 2

```

8 12
FD EMPFILE.
01 EMP-REC.
   02 EMP-NO          PIC X(5).
FD BFILE.
01 DEP-REC.
   02 EMP-NO          PIC X(5).
01 PAY-REC.
   02 EMP-NO          PIC X(5).
DEFINITION DIVISION.
DEFINED FILE EMPLS
DEFINED RECORD EMPLOYEE FROM EMP-REC
  IDENTIFIER EMP-NO
SUPPLEMENT DEPENDENT FROM DEP-REC
  POINTER IS EMP-NO
  FILL KEY TO 'D'
SUPPLEMENT PAYROLL FROM PAY-REC
  FILL KEY TO 'P'

```

In this example, the source of record EMPLOYEE is EMP-REC in indexed file EMPFILE. The sources of supplements DEPENDENT and PAYROLL are located in a separate indexed file (BFILE). By specifying the FILL KEY clause, you can distinguish between the records. You only need the POINTER clause for the first supplement, DEPENDENT.

3.35. SPECIFYING THE EFFECTS OF DEFINED RECORD CHANGES (ROLE IN UPDATE CLAUSE)

Purpose

Use the ROLE IN UPDATE clause to specify how the supplement's source affects or is affected by adding, deleting, or changing a defined record. The format is:

Format

```

ASSUMES { CONTROLLING } ROLE IN UPDATE
        { CONTROLLED }
        { NEUTRAL   }

```

Options

Table 3-3 summarizes the ROLE IN UPDATE options.

SUPPLEMENT DEFINITION

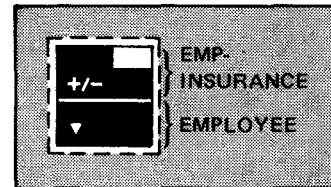
Table 3-3. ROLE IN UPDATE Options and Their Meanings

| ROLE IN UPDATE | Adding a Defined Record | Deleting a Defined Record | Changing a Defined Record |
|------------------|--|--|--|
| CONTROLLING ▽ | Supplement's source must already be in the file. | No effect on supplement's source. | Supplement's source cannot be changed. |
| CONTROLLED △ | Supplement's source is also added to the file. | Supplement's source is also deleted from the file. | Supplement's source can be changed. |
| NEUTRAL ⌵ | No effect. | No effect on supplement's source. | Supplement's source cannot be changed. |

Specifying CONTROLLING

Function

CONTROLLING means that you cannot add a defined record unless the source of the supplement is already in the file.



Restrictions

It does not control deletion of a defined record. When you add, delete, or change a defined record, the supplement's source is not affected. If you specify MUST ADD (3.39) or ALLOW CHANGE (3.40) for any supplement item, the processor issues error messages and won't successfully create a data definition record.

Example

```

8  12
FD  INSFILE
01  INS-REC.
01  EMP-REC.
DEFINITION DIVISION.
DEFINED FILE INSURANCE
DEFINED RECORD EMP-INSURANCE FROM INS-REC
      ALLOW ADD AND DELETE
SUPPLEMENT EMPLOYEE FROM EMP-REC
      ASSUMES CONTROLLING ROLE IN UPDATE

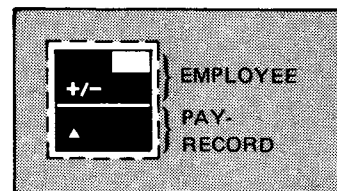
```

In this example, defined record EMP-INSURANCE is from INS-REC. Supplement EMPLOYEE, from EMP-REC, contains data pertaining to employee insurance records. Because you use the CONTROLLING option, you can add an employee insurance record only when employee data is available. You cannot change the value of any items contained in supplement EMPLOYEE.

Specifying CONTROLLED

Function

CONTROLLED means that you add or delete this supplement's source whenever you add or delete a defined record. Thus, the primary part and its supplement are added and deleted as a pair. The supplement's source must not be a repeating group item.



Restrictions

You can specify MUST ADD or ALLOW CHANGE for a supplement item only when you specify CONTROLLED for the supplement. When this supplement's source is in the file before you add a record, a new source occurrence replaces the old.

Alternate access

You can also use the CONTROLLED option to access the same data in a different way when the supplement's source is also the source of the primary part in a different defined file. See 4.5 for an example.

Example

```

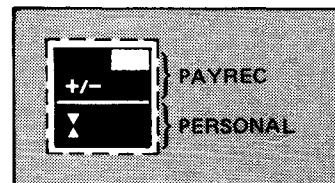
8  12
FD  PAYFILE
01  EMP-REC.
01  PAY-REC.
DEFINED FILE PAYROLL
DEFINED RECORD EMPLOYEE FROM EMP-REC
      ALLOW ADD AND DELETE
SUPPLEMENT PAY-RECORD FROM PAY-REC
      ASSUMES CONTROLLED
  
```

Defined record EMPLOYEE is from EMP-REC. Supplement PAYROLL, from PAY-REC, contains employee payroll data. Because you specify the CONTROLLED option, when you add or delete employee data, you must add or delete payroll data. You can also change the value of any items contained in supplement PAYROLL.

Specifying NEUTRAL

Function

NEUTRAL means that this supplement's source neither affects nor is affected by adding or deleting a defined record. You can add a defined record without the supplement's source already being in the file. And, when you add or delete a defined record, the supplement's source does not change.



SUPPLEMENT DEFINITION**Restrictions**

You use the data in the supplement for retrieval purposes only, and you cannot change the value of any supplement item. If you specify **MUST ADD** or **ALLOW CHANGE** for a supplement item, the processor issues error messages and won't successfully create a data definition record. **NEUTRAL** is selected by default when you omit the **ROLE IN UPDATE** clause.

Example

```

8 12
FD PAYFILE.
Ø1 EMPLOYEE-PAY.
Ø1 PERSONAL-DATA.
DEFINED FILE PAYROLL
DEFINED RECORD PAYREC FROM EMPLOYEE-PAY
    ALLOW ADD AND DELETE
SUPPLEMENT PERSONAL FROM PERSONAL-DATA
    ASSUMES NEUTRAL

```

Defined record **PAYREC** is from **EMPLOYEE-PAY**. Supplement **PERSONAL**, from **PERSONAL-DATA**, contains personal data about employees, such as health and job history information. Because you specify the **NEUTRAL** option, the addition or deletion of an employee's pay record does not affect and is not affected by the status of his personal record. Data in his personal record is used for information only; it cannot be changed. You obtain the same effect by omitting the **ROLE IN UPDATE** clause.

3.36. ITEM DEFINITION**No IDENTIFIER
statements**

A supplement item definition follows the same format as a defined record item definition (See 3.19 through 3.28). However, a supplement item definition cannot contain **IDENTIFIER** statements; it can only contain **ITEM** statements.

Figure 3-26 shows the format of the supplement item definition.

Format

```

ITEM [item-name-2 FROM] data-name-2
[HIDDEN]
[MUST ADD]
[ALLOW CHANGE]
[VALUE IS ] literal-1 [THROUGH] literal-2 ]
[VALUES ARE ] [THRU ]
[.literal-3 [THROUGH] literal-4] ... ]
[ALSO [item-alias-1 FROM] item-name-3
[. [item-alias-2 FROM] item-name-4] ... ]

```

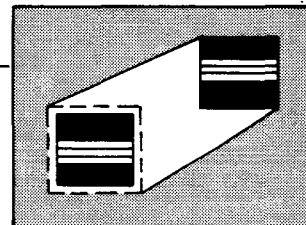
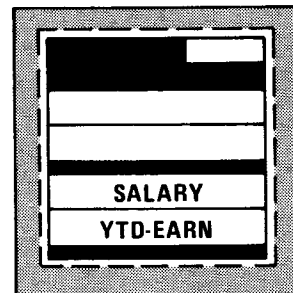


Figure 3-26. Item Definition Format for a Supplement

3.37. NAMING THE ITEM (ITEM CLAUSE)

Format and rules

The ITEM clause follows the same format and rules in a supplement as in a defined record (see 3.24), except that data-name-2 is a data division data name that comes from the supplement's source.



Example

```

8 12
DATA DIVISION.
FILE SECTION.
FD EMPFILE.
  01 EMPLOYEE-REC.
    02 DEPT-NR          PIC X(4).
    02 EMP-NM          PIC X(21).
    02 EMP-NR          PIC X(5).
  01 PAY-REC.
    02 SALARY          PIC 9(5)V99.
    02 YTD-EARNINGS    PIC 9(7)V99 USAGE COMP-3.
DEFINITION DIVISION.
DEFINED FILE PAYROLL
DEFINED RECORD EMPLOYEE
  FROM EMPLOYEE-REC
  IDENTIFIER DEPT-NR
  ITEM EMPL-NAME FROM EMP-NM
  ITEM EMP-NR
SUPPLEMENT PAYDATA
  FROM PAY-REC
  ITEM SALARY
  ITEM YTD-EARN FROM YTD-EARNINGS

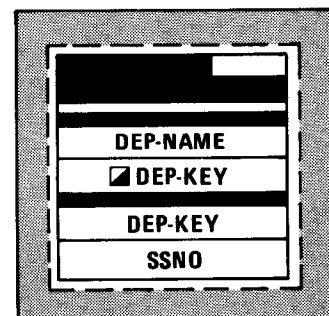
```

Supplement PAYDATA includes items SALARY and YTD-EARN in defined record EMPLOYEE.

3.38. PREVENTING ITEM DISPLAY (HIDDEN OPTION)

Format and rules

The HIDDEN option follows the same format and rules in a supplement as in a defined record. (See 3.25).



SUPPLEMENT DEFINITION*Example*

```

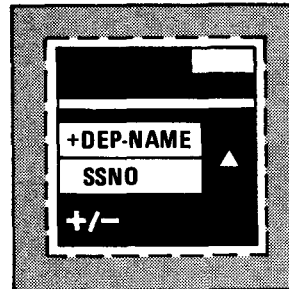
8 12
-----
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
  IDENTIFIER EMP-NR
  ITEM EMP-NM
SUPPLEMENT DEP1 FROM DEPENDENT-REC
  ITEM DEP-NAME
  ITEM DEP-KEY HIDDEN
SUPPLEMENT DEP2 FROM SUPP-REC
  POINTER IS DEP-KEY
  ITEM SSNO

```

In this example, each DEP1 record gives a pointer (DEP-KEY) that locates DEP2 records. You can use the HIDDEN option when you don't want DEP-KEY data displayed at a UNIQUE terminal. Supplements DEP1 and DEP2 include DEP-NAME and SSNO in defined record EMPLOYEE.

3.39. SPECIFYING A REQUIRED ITEM (MUST ADD OPTION)*Format and rules*

The MUST ADD option follows the same format and rules in a supplement as in a defined record. (See 3.26.) The MUST ADD option applies only when the supplement's ROLE IN UPDATE (3.35) is CONTROLLED.

*Example*

```

8 12
-----
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
  IDENTIFIER EMP-NR
  ITEM EMP-NM
SUPPLEMENT DEPENDENT FROM DEPENDENT-REC
  ASSUMES CONTROLLED ROLE IN UPDATE
  ITEM DEP-NAME MUST ADD
  ITEM SSNO

```

Before you add an EMPLOYEE record, supplement item DEP-NAME must contain a valid value. Supplement DEPENDENT's ROLE IN UPDATE must be CONTROLLED.

3.40. ALLOWING CHANGES TO THE ITEM (ALLOW CHANGE OPTION)

Format and rules

The ALLOW CHANGE option follows the same format and rules in a supplement as in a defined record. (See 3.27.) The ALLOW CHANGE option applies only when the supplement's ROLE IN UPDATE (3.35) is CONTROLLED.

| | |
|------------|--|
| → DEP-NAME | |
| SSNO | |
| → AGE | |
| +/- | |

Example

```

8 12
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
  ALLOW ADD AND DELETE
  IDENTIFIER EMP-NR
  ITEM EMP-NM
SUPPLEMENT DEPENDENT FROM DEPENDENT-REC
  ASSUMES CONTROLLED
  ITEM DEP-NAME ALLOW CHANGE
  ITEM SSNO
  ITEM AGE ALLOW CHANGE

```

In this example, you can change items DEP-NAME and AGE in supplement DEPENDENT. You cannot change item SSNO. To change any supplement item, supplement DEPENDENT's ROLE IN UPDATE must be CONTROLLED.

3.41. SPECIFYING A VALUE RANGE FOR THE ITEM (VALUE CLAUSE)

Format and rules

The VALUE clause follows the same format and rules in a supplement as in an ITEM statement for a defined record. (See 3.28.) The VALUE clause applies only when you specify the supplement's ROLE IN UPDATE (3.35) as CONTROLLED.

| | |
|-------------|---|
| HOURLY-RATE | ▼ |
| → =0425- | |
| 1500 | |
| +/- | |

SUPPLEMENT DEFINITION*Example*

```

8 12
-----
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
  ALLOW ADD AND DELETE
  IDENTIFIER EMP-NR
  ITEM EMP-NM
SUPPLEMENT PAYDATA FROM PAY-REC
  ASSUMES CONTROLLED
  ITEM HOURLY-RATE ALLOW CHANGE VALUE IS 0425 THROUGH 1500

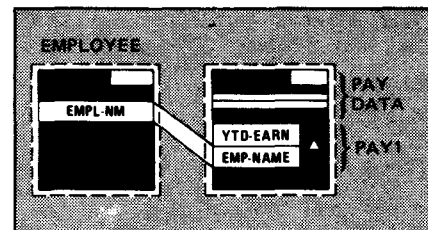
```

You can change supplement item HOURLY-RATE, but, unless the new values fall between 425 and 1500, IMS rejects the update and returns an invalid request indicator (003) in the program status code.

3.42. INCLUDING PREVIOUSLY DEFINED ITEMS IN THE DEFINED RECORD (ALSO CLAUSE)

Purpose

With the ALSO clause, the current defined record can include items from its direct ancestors. You do not need to write supplement definitions for items named in an ALSO clause, but the clause must follow any supplement definitions you do write.

*Format and rules*

This clause uses the same format and rules as when it follows a defined record's item definitions. (See 3.29.)

Example

```

8 12
-----
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
  IDENTIFIER JOB-NR
  ITEM EMPL-NM
DEFINED RECORD PAYDATA FROM PAY-REC
  PARENT IS EMPLOYEE
  IDENTIFIER DEPT-NR
  ITEM EMP-NR
  ITEM SALARY
SUPPLEMENT PAY1 FROM SUPP-REC
  ASSUMES CONTROLLED
  ITEM YTD-EARN
  ALSO EMP-NAME FROM EMPL-NM

```

The ALSO clause includes item EMP-NAME in defined record PAYDATA. EMP-NAME was previously defined as item EMPL-NM in ancestor record EMPLOYEE. This clause follows the supplement definition used to include another item, YTD-EARN, in PAYDATA.

3.43. SUBRECORD DEFINITION

Purpose

You can include two or more variations of a defined record in the same data definition. A variation of a defined record is a subrecord.

Contents

A subrecord contains the same data as the defined record, but it may differ in:

- the number, position, and names of items included; and
- the update functions allowed.

Accessed through subfiles

You can access a subrecord only by including it in a subfile. Terminal operators using UNIQUE must name the subfile in the OPEN command before requesting a subrecord.

Rule

You must write a separate subrecord definition for each variation of the defined record. Figure 3-27 shows the format of the subrecord definition.

Format

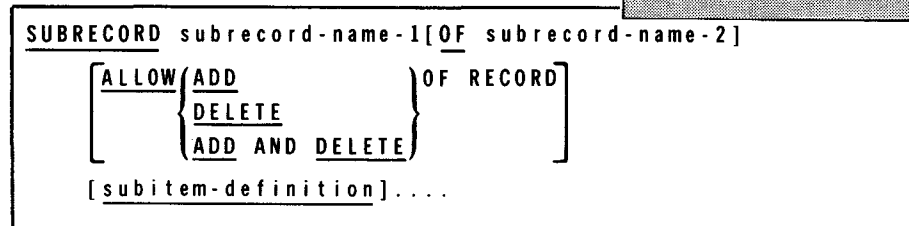





Figure 3-27. Subrecord Definition Format

Identifiers

Each subrecord automatically includes all identifier items from the defined record. Therefore, you only use the IDENTIFIER clause to change an identifier name. You include other items in the subrecord by writing subitem definitions (3.47 through 3.52).

SUBRECORD DEFINITION

Subrecord independence Although a subrecord is a subset of a defined record, a defined record and its subrecord are independent of each other. Thus, you can allow updates:

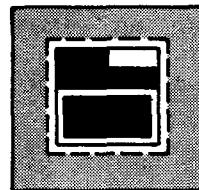
| | |
|---|--|
|  | to the defined record and not to the subrecord; |
|  | to the subrecord and not to the defined record; or |
|  | to both the defined record and its subrecord. |

Subitem independence The same rule applies to clauses you specify for defined record items and subitems in a subrecord.

3.44. NAMING THE SUBRECORD (SUBRECORD CLAUSE)

Function The SUBRECORD clause begins a subrecord definition and names the subrecord. Starting in column 8, the format is:

Format SUBRECORD subrecord-name-1



Subrecord-name-1 Subrecord-name-1 is a 1- to 30-character name, unique within the data definition, that identifies the subrecord.

Example

```

8  12
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
SUBRECORD EMPLOYEE-SUB1

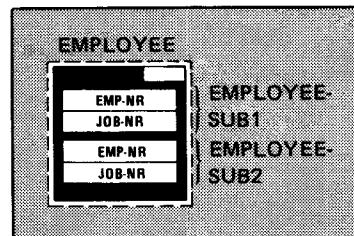
```

This example identifies EMPLOYEE-SUB1 as a subrecord of defined record EMPLOYEE.

3.45. INCLUDING PREVIOUSLY DEFINED ITEMS IN THE SUBRECORD (OF CLAUSE)

Reducing subitem definitions

You can use the OF clause to reduce the number of subitem definitions (3.47 through 3.52) you write when you have already used other subrecord definitions in a defined record.



Purpose

The OF clause lets you include subitems already defined in a previous subrecord without redefining them.

The format is:

Format

OF subrecord-name-2

Subrecord-name-2

Subrecord-name-2 is a subrecord already defined within this defined record definition.

Example

```

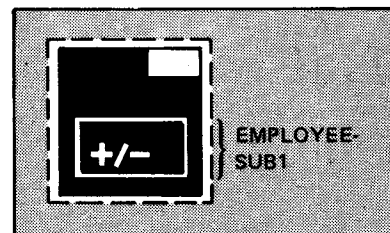
8 12
-----
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
SUBRECORD EMPLOYEE-SUB1
    ITEM EMP-NR
    ITEM JOB-NR
SUBRECORD EMPLOYEE-SUB2 OF EMPLOYEE-SUB1
    
```

The OF clause in subrecord EMPLOYEE-SUB2 says that subitems EMP-NR and JOB-NR, described in subrecord EMPLOYEE-SUB1's definition, are also contained in EMPLOYEE-SUB2.

3.46. ALLOWING SUBRECORD ADDITIONS AND DELETIONS (ALLOW ADD AND DELETE CLAUSE)

Purpose

The ALLOW ADD AND DELETE clause permits terminal operators using UNIQUE or your action programs to add or delete subrecord occurrences. The format is:






Format

ALLOW { ADD
DELETE
ADD AND DELETE } OF RECORD

SUBRECORD DEFINITION

Parameters

| | | |
|----------------|---------------------------------|--|
| ADD | Allows only additions. |  |
| DELETE | Allows only deletions. |  |
| ADD AND DELETE | Allows additions and deletions. |  |

Adding and deleting records

Without any of these statements in a subrecord definition, you cannot add or delete occurrences of the subrecord. For example, when you do not include ALLOW ADD, ALLOW DELETE, or ALLOW ADD AND DELETE in the subrecord definition, IMS rejects as invalid:

- any input of the UNIQUE ADD or DELETE commands; or
- any add or delete function issued by an action program.

Example

```

8  12
SUBRECORD EMPLOYEE-SUB1
  ALLOW ADD AND DELETE

```

This allows you to add or delete occurrences of subrecord EMPLOYEE-SUB1 in the defined file.

3.47. SUBITEM DEFINITION*Function*

The subitem definition is part of the subrecord definition. It includes in the subrecord items previously described in the defined record's item definitions.

Figure 3-28 shows the format of the subitem definition.

Write a subitem definition for each item in the subrecord.

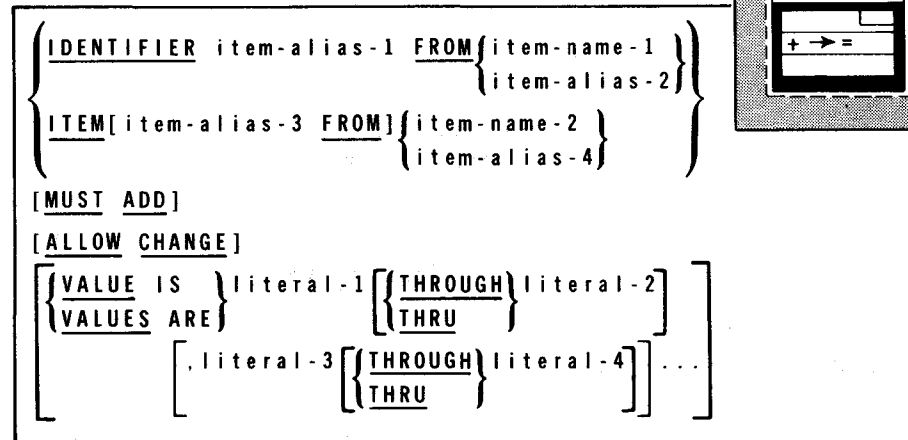
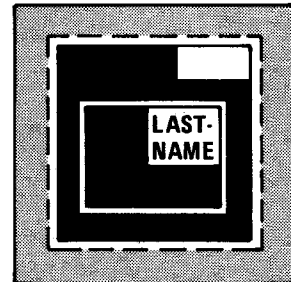
Format

Figure 3-28. Subitem Definition Format

3.48. NAMING THE IDENTIFIER (IDENTIFIER CLAUSE)**Purpose**

The subrecord automatically includes identifier items, but you can use an IDENTIFIER clause to change the name of an identifier. The format is:

**Format**

```
IDENTIFIER item-alias-1 FROM {item-name-1}
{item-alias-2}
```

Item-alias-1

Item-alias-1 is a 1- to 30-character name that renames the identifier item. When a terminal operator accesses this subrecord through UNIQUE, this name is displayed as a terminal column header.

Item-name-1

Item-name-1 is an identifier item previously defined within this defined record definition. Use this option when the subrecord definition does not include an OF clause (3.45).

Item-alias-2

Item-alias-2 is an identifier item already renamed in a previous subrecord definition. Use this option when the current subrecord includes the OF clause. In that OF clause, you must specify that previous subrecord as subrecord-name-2.

SUBRECORD DEFINITION*Example*

```

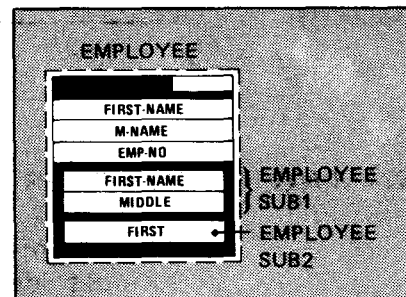
8 12
-----
DEFINED RECORD EMPLOYEE
  IDENTIFIER L-NAME
  ITEM FIRST-NAME
  ITEM M-NAME
  ITEM EMP-NO
SUBRECORD EMPLOYEE-SUB1
  IDENTIFIER LAST-NAME FROM L-NAME

```

Identifier L-NAME in defined record EMPLOYEE is renamed LAST-NAME in subrecord EMPLOYEE-SUB1.

3.49. NAMING THE ITEM (ITEM CLAUSE)*Function*

The ITEM clause includes in the subrecord an item described in a preceding defined record or subrecord. It can also give a new name (alias) to the item for use within this subrecord. The format is:

*Format*

```

ITEM [item-alias-3] FROM { item-name-2 }
                          { item-alias-4 }

```

Item-alias-3

Item-alias-3 names the subitem. It must be 1 to 30 characters, unique within the subrecord definition. When a terminal operator accesses this subrecord through UNIQUE, this name is displayed as a terminal column header. When the subitem has the same name as item-name-2 or item-alias-4, omit item-alias-3 and the word FROM.

Item-name-2

Item-name-2 is an item previously defined within this defined record definition. Use this option when this subrecord definition does not include an OF clause (3.45).

Item-alias-4

Item-alias-4 is a subitem already renamed in a previous subrecord definition. Use this option when the current subrecord includes the OF clause. In that OF clause, you must specify that previous subrecord as subrecord-name-2.

Example

```

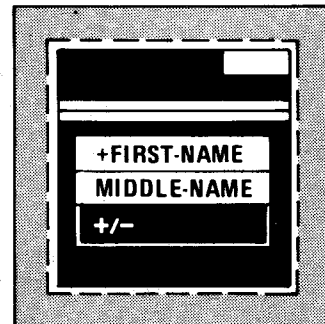
8 12
-----
DEFINED RECORD EMPLOYEE
  IDENTIFIER LAST-NAME
  ITEM FIRST-NAME
  ITEM M-NAME
  ITEM EMP-NO
SUBRECORD EMPLOYEE-SUB1
  ITEM FIRST-NAME
  ITEM MIDDLE FROM M-NAME
SUBRECORD EMPLOYEE-SUB2 OF EMPLOYEE-SUB1
  ITEM FIRST FROM FIRST-NAME

```

The items FIRST-NAME and M-NAME are defined in record EMPLOYEE. In subrecord EMPLOYEE-SUB1, FIRST-NAME retains the same name and M-NAME is renamed MIDDLE. EMPLOYEE-SUB1 automatically includes identifier item LAST-NAME. EMPLOYEE-SUB2 includes identifier item LAST-NAME, item MIDDLE, and item FIRST (renamed from item FIRST-NAME in subrecord EMPLOYEE-SUB1).

3.50. SPECIFYING A REQUIRED ITEM (MUST ADD OPTION)*Purpose*

You can use the MUST ADD option to specify that a certain subitem must be present and contain a valid value before a terminal operator can add a subrecord occurrence. The format is:

*Format*

MUST ADD

Using the MUST ADD option

To be valid, a numeric subitem must be nonzero, and an alphanumeric subitem must contain other than all spaces.

Restrictions

MUST ADD works only when you specify the ALLOW ADD or ALLOW ADD AND DELETE clause in your subrecord definition.

Example

```

8 12
-----
DEFINED RECORD EMPLOYEE FROM EMPLOYEE-REC
  IDENTIFIER LAST-NAME
  ITEM FIRST-NAME
  ITEM M-NAME
SUBRECORD EMPLOYEE-SUB1
  ALLOW ADD AND DELETE
  ITEM FIRST-NAME MUST ADD
  ITEM MIDDLE-NAME FROM M-NAME

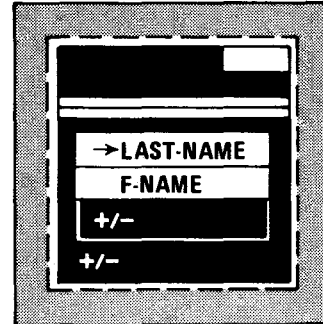
```

SUBRECORD DEFINITION

Before you add an EMPLOYEE-SUB1 subrecord occurrence, subitem FIRST-NAME must contain a valid value. The MUST ADD option is not specified for MIDDLE-NAME, so you do not have to include it to add an EMPLOYEE-SUB1 subrecord.

3.51. ALLOWING CHANGES TO THE ITEM (ALLOW CHANGE OPTION)*Purpose*

Terminal operators can only make changes to the subitem when you specify the ALLOW CHANGE option.



The format is:

Format

ALLOW CHANGE

Example

```

8 12
DEFINED RECORD EMPLOYEE
  ALLOW ADD AND DELETE
  IDENTIFIER EMP-NO
  ITEM LAST-NAME
  ITEM FIRST-NAME
SUBRECORD EMPLOYEE-SUB1
  ALLOW ADD AND DELETE
  ITEM LAST-NAME ALLOW CHANGE
  ITEM F-NAME FROM FIRST-NAME

```

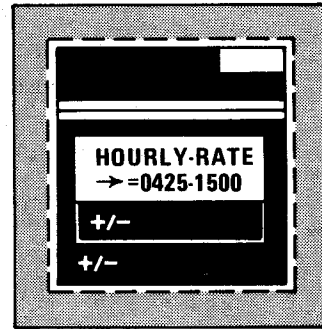
In this example, ALLOW CHANGE specifies that you can change item LAST-NAME in subrecord EMPLOYEE-SUB1. You cannot change item FIRST-NAME because ALLOW CHANGE is not specified.

*Effect of omitting
ALLOW CHANGE*

Without ALLOW CHANGE, IMS won't carry out any requested subitem value changes to records on disk. When an action program attempts to update a record containing the changed item value, IMS returns control to the action program with an invalid request indicator (003) in the program status code.

3.52. SPECIFYING A VALUE RANGE FOR THE ITEM (VALUE CLAUSE)*Purpose*

You can use the VALUE clause to specify the valid value ranges a subitem can have for it to be added or changed. When you omit the VALUE clause, IMS accepts any value consistent with the PICTURE or USAGE clauses specified for this item's source (see Figure 3-2).



The format is:

Format

```
{ VALUE IS } literal-1 { THROUGH } literal-2
{ VALUES ARE } [ literal-3 { THROUGH } literal-4 ] ...
```

Literal-1, literal-2, ...

Literal-1, literal-2,... specify the values or value ranges allowed for a subitem being added or changed. You must put the values for literal-1, literal-2, etc, in ascending order and can specify no more than 64 literals. Alphanumeric literals have to be enclosed in single quotes; numeric literals do not.

Example

```
8 12
DEFINED RECORD PAYROLL
  ALLOW ADD AND DELETE
  IDENTIFIER EMP-NR
  ITEM EMP-NM
SUBRECORD PAYROLL-SUB1
  ALLOW ADD AND DELETE
  ITEM HOURLY-RATE ALLOW CHANGE VALUE IS 0425 THROUGH 1500
```

You can change subitem HOURLY-RATE, but, unless the new values fall between 425 and 1500, IMS rejects the update.

SUBFILE DEFINITION

3.53. SUBFILE DEFINITION

Purpose You can include two or more variations of the defined file in the same data definition by using a subfile. A subfile definition describes an independent subset of a defined file. A subfile contains defined records, subrecords, or both. To access subrecords, you must include them in a subfile definition.

Contents Defined files and subfiles can differ in the number and makeup of the defined record types they contain. Each subfile definition can define only one variation of the defined file.

Figure 3-29 shows the format of the subfile definition.

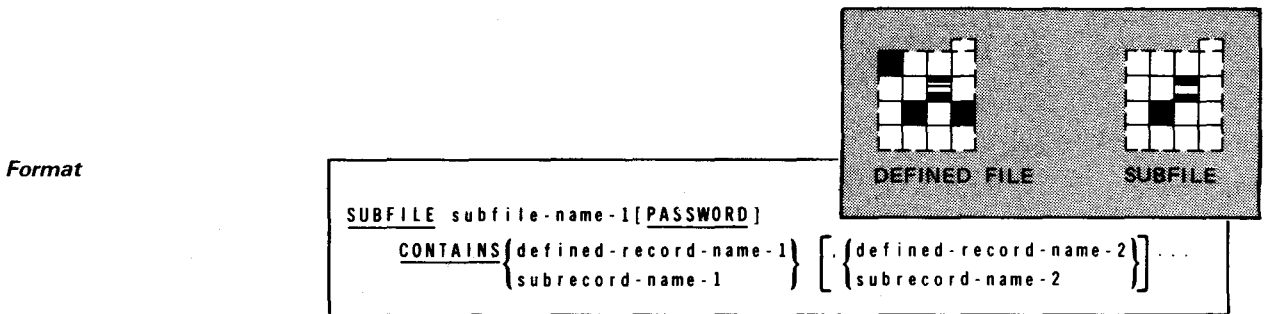
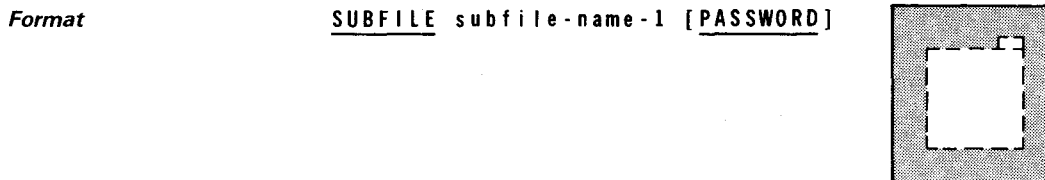


Figure 3-29. Subfile Definition Format

3.54. NAMING THE SUBFILE (SUBFILE STATEMENT)

Function The SUBFILE statement begins a subfile definition and names the subfile. Starting in column 8, the format is:



Subfile-name-1 Subfile-name-1 is one to seven characters and must differ from the names of:

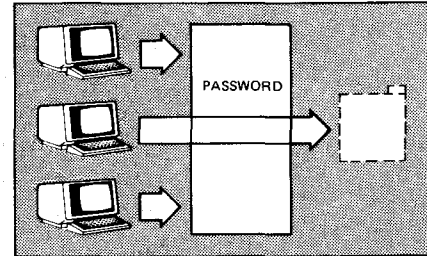
- the defined file and other subfiles within the data definition; and
- any conventional file assigned to IMS.

Using subfile names

Within the data definition, subfile names are used the same way as a defined file name.

Passwords

With UNIQUE, you allow access to subfiles by using the PASSWORD clause; with action programs, you do not use this clause. When you specify PASSWORD, terminal operators enter subfile-name-1 as a password in the UNIQUE OPEN command to access a subfile.

**PASSWORD clause****Defining passwords with the NAMEREC utility**

You can omit PASSWORD and define a password with the NAMEREC file utility. This allows you to limit subfile access to specific terminals and use multiple passwords to access the same subfile. A password defined in the NAMEREC utility does not cancel one defined in the data definition unless the passwords are the same. The IMS system support functions user guide, UP-8364 (current version) describes password definition with the NAMEREC utility.

Effect of omitting password definition

You must define a password using either the PASSWORD clause or the NAMEREC utility; otherwise, terminal operators using UNIQUE cannot access the subfile.

Outside references

You use the subfile name to refer to the subfile in a number of places outside the data definition:

- keyword parameter DFILE in the ACTION section of the configuration;
- keyword parameter FN in the password definition input to the NAMEREC file utility;
- the defined-file-name parameter in action program function calls to defined record management; and
- the defined-file-name field in the program information block for COBOL, BAL, and RPG II action programs.

The IMS system support functions user guide, UP-8364 (current version) describes configuration and the NAMEREC file utility. Action programs are discussed in the current versions of the IMS action programming in COBOL and basic assembly language (BAL) user guide, UP-9207 and the IMS action programming in RPG II user guide, UP-9206.

SUBFILE DEFINITION*Example*

```

8 12
-----
DEFINED FILE JOBFILE PASSWORD
DEFINED RECORD EMPLOYEE
DEFINED RECORD PAYDATA
SUBFILE EMPFILE PASSWORD
SUBFILE PAYFILE

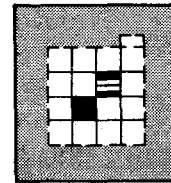
```

Defined file JOBFILE has two record types, EMPLOYEE and PAYDATA. To restrict access to PAYDATA, you define two subfiles, EMPFILE and PAYFILE. All terminal operators can access EMPFILE by using the subfile name EMPFILE as the password. On the other hand, only those terminals named in a NAMEREC utility password definition can access PAYFILE.

3.55. IDENTIFYING RECORDS INCLUDED IN THE SUBFILE (CONTAINS CLAUSE)

Purpose

Use the CONTAINS clause to name the defined records and subrecords included in this subfile. The format is:

*Format*

```

CONTAINS { defined-record-name-1 } [ { defined-record-name-2 } ] ...
          { subrecord-name-1 }   { subrecord-name-2 }

```

Defined-record-name-1

Defined-record-name-1, defined-record-name-2,... name defined records included in this subfile.

Subrecord-name-1

Subrecord-name-1, subrecord-name-2,... name subrecords included in this subfile.

Coding rules

You can include only one entry for each defined record or any of its subrecords. Put entries in the same order as their corresponding defined record definitions appear in the data definition. Before you submit an entry for a defined record in a hierarchical file, you must have an entry for each of that record's direct ancestors.

Example

```

8 12
-----
DEFINED FILE PAYROLL
DEFINED RECORD EMPLOYEE
SUBRECORD EMPLOYEE-SUB1
SUBFILE EMPFILE
      CONTAINS EMPLOYEE-SUB1

```

Subfile EMPFILE consists of only one type of defined record, known by its subrecord name, EMPLOYEE-SUB1.

4. Data Definition Examples

This section contains extended examples of how to use the data definition language. It includes examples of simple and hierarchical defined files, supplements, subfiles, and interrelated defined files.

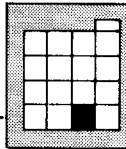
EXAMPLE OF SIMPLE DEFINED FILE

4.1. EXAMPLE OF A SIMPLE DEFINED FILE

- Preview of example* Figures 4-1 through 4-5 show the simple defined file EMPLS.
- Source file* EMPLS comes from indexed file EMPFILE, whose first few records are shown in Figure 4-1. Each record begins with a 21-character field that contains its record key. The record is EMPLOYEE-REC, and the key is EMPL-NAME.
- Data definition* Figure 4-2 shows the data definition for defined file EMPLS.
- Records delivered* Figure 4-3 shows the first few EMPLS records UNIQUE lists at a terminal in response to a LIST command; Figure 4-4 shows how IMS delivers them to an action program. Each record contains identifier item EMP-NM and two other items, SSNO and DEPT-NAME. With UNIQUE, these item names appear as column headers at terminals.
- Action program record areas* To access defined file EMPLS, you must provide a place in your action program to receive defined record EMPLOYEE. Figure 4-5 shows COBOL, BAL, and RPG II action program descriptions of the record areas for receiving EMPLOYEE. In addition to the items in the defined record, you must describe a status byte for each item. (See 5.4.) The prefix 'S-' in the COBOL action program corresponds to the prefix the data definition processor generates for each item status byte data name.

NOTE:

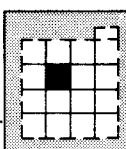
The data definition processor changes the level numbers in the COBOL description because it provides an I/O area for the defined file.



```

ABRAMS, MARK^^^^^^^^^^309314828PRODUCTION^^^^45189
ACKERMAN, GWEN^^^^^^^^229587259MARKETING^^^^22041
ADAMS, BEN^^^^^^^^^^242890344MARKETING^^^^48297
ALLEN, JAMES^^^^^^^^^^197448473OFFICESERVICES15548
ARTHUR, MONICA^^^^^^^^067531675LEGAL^^^^^^^^10001
BAINES, CHARLES^^^^^^189634798ACCOUNTING^^^^32342
BILKER, HAROLD^^^^^^348978885PRODUCTION^^^^57690
BONDS, ALLISON^^^^^^228561194QUALITYCONTROL40251
BROOKS, ELAINE^^^^^^177338959DEVELOPMENT^^23866
BROOKS, JOHN^^^^^^129312210MARKETING^^^^27454
  
```

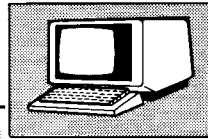
Figure 4-1. Excerpt from EMPFILE, an Indexed Employee File



```

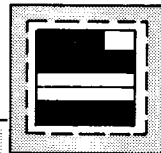
LINC NO.  SEQ.      SOURCE STATEMENT
00001      IDENTIFICATION DIVISION.
00002      PROGRAM-ID. BASIC-DATA-DEF.
00003      DATA DIVISION.
00004      FILE SECTION.
00005      FD EMPFILE.
00006      D1 EMPLOYEE-REC.
00007         02 EMPL-NAME           PIC X(21).
00008         02 SSNO                 PIC X(9).
00009         02 DEPT-NAME            PIC X(21).
00010         02 ENTRY                 PIC X(5).
00011      DEFINITION DIVISION.
00012      DEFINED FILE EMPLS PASSWORD
00013      DEFINED RECORD EMPLOYEE
00014      FROM EMPLOYEE-REC
00015      ALLOW ADD AND DELETE
00016      IDENTIFIER EMP-NM FROM EMPL-NAME
00017      ITEM SSNO
00018      ITEM DEPT-NAME
                                } Key is EMP-NM
  
```

Figure 4-2. Data Definition for Defined File EMPLS



| EMP - NM | SSNO | DEPT - NAME |
|-----------------|-----------|----------------|
| ABRAMS, MARK | 309314828 | PRODUCTION |
| ACKERMAN, GWEN | 229587259 | MARKETING |
| ADAMS, BEN | 242890344 | MARKETING |
| ALLEN, JAMES | 197448473 | OFFICESERVICES |
| ARTHUR, MONICA | 067531675 | LEGAL |
| BAINES, CHARLES | 189634798 | ACCOUNTING |
| BILKER, HAROLD | 348978885 | PRODUCTION |
| BONDS, ALLISON | 228561194 | QUALITYCONTROL |
| BROOKS, ELAINE | 177338959 | DEVELOPMENT |
| BROOKS, JOHN | 129312210 | MARKETING |

Figure 4-3. First Few EMPLS Records, as Listed at a Terminal by UNIQUE



| | | |
|-----------------|-----------|----------------|
| ABRAMS, MARK | 309314828 | PRODUCTION |
| ACKERMAN, GWEN | 229587259 | MARKETING |
| ADAMS, BEN | 242890344 | MARKETING |
| ALLEN, JAMES | 197448473 | OFFICESERVICES |
| ARTHUR, MONICA | 067531675 | LEGAL |
| BAINES, CHARLES | 189634798 | ACCOUNTING |
| BILKER, HAROLD | 348978885 | PRODUCTION |
| BONDS, ALLISON | 228561194 | QUALITYCONTROL |
| BROOKS, ELAINE | 177338959 | DEVELOPMENT |
| BROOKS, JOHN | 129312210 | MARKETING |

Figure 4-4. First Few EMPLS Records, as Delivered to an Action Program

EXAMPLES OF HIERARCHICAL DEFINED FILES

4.2. EXAMPLES OF HIERARCHICAL RECORDS IN DEFINED FILES*Preview of examples*

Figures 4-6 through 4-10 show parts of defined files and the data definitions needed to define:

- an indexed file containing two record types;
- an indexed file containing a repeating group; and
- two indexed files.

Different descriptions of same defined file

In these examples, three different data definitions describe the same defined file. Although the defined file comes from three sources that differ in content and organization, there is no difference in:

- the defined file delivered to the action program; or
- the appearance of the defined file at the terminal when it is accessed through UNIQUE.

Reorganize/redefine files

You can reorganize source files and redefine defined files without changing your action programs or terminal operating procedures.

Records delivered and action program record areas

Figures 4-11 and 4-12 show, for all three examples, how the defined file appears at a terminal through UNIQUE and how it is delivered to an action program. Figure 4-13 shows COBOL and BAL action program descriptions of record areas receiving the defined records.

Parent-Child Defined Records Using Several Record Types as a Source*Source file*

Figure 4-6 shows the first few records in indexed file EMPFILE that defined records EMPLOYEE and DEPENDENT come from. Figure 4-7 shows the data definition for defined file EMPLOYEES, which contains hierarchical records EMPLOYEE and DEPENDENT. The order of the records in EMPLOYEES matches the order of their sources in EMPFILE.

Parent-child relationship

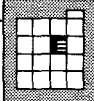
There are several ways that parent and child defined records can be related in files. (See 3.13.) In this example, the source (DEPENDENT-REC) of child record DEPENDENT directly follows the source (EMPLOYEE-REC) of DEPENDENT's parent record, EMPLOYEE.

EXAMPLES OF HIERARCHICAL DEFINED FILES

Parent-Child Defined Records Using a Repeating Group as a Source

Source file

Indexed file EMP-RG, shown in Figure 4-8, contains the same information as indexed file EMPFILE in Figure 4-6, but it is organized differently. A table within each employee record contains the dependent information.



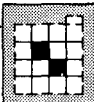
```

ABRAMS, MARK^^^^^^^^^^^^^^^^^PRODUCTION^^^^^01
ABRAMS, JOAN^^^^^^^^^^^^^^^^^326090119
ACKERMAN, GWEN^^^^^^^^^^^^^^^MARKETING^^^^^00
ADAMS, BEN^^^^^^^^^^^^^^^^^MARKETING^^^^^02
ADAMS, BONNIE^^^^^^^^^^^^^^^^^298345429
ADAMS, SARAH^^^^^^^^^^^^^^^^^395006297
ALLEN, JAMES^^^^^^^^^^^^^^^^^OFFICESERVICES00
ARTHUR, MONICA^^^^^^^^^^^^^^^LEGAL^^^^^^^^^02
ARTHUR, WAYNE^^^^^^^^^^^^^^^^^120862348
BAINES, CHARLES^^^^^^^^^^^^^^^ACCOUNTING^^^^03
BAINES, CLAUDIA^^^^^^^^^^^^^^^631765514
BAINES, JOSEPH^^^^^^^^^^^^^^^622110963
BAINES, LESLIE^^^^^^^^^^^^^^^489259906
BILKER, HAROLD^^^^^^^^^^^^^^^PRODUCTION^^^^00
  
```

Figure 4-8. Excerpt from EMP-RG, an Indexed File Containing a Repeating Group

Data definition

Figure 4-9 shows the data definition for defined file EMPLOYEES when the source is a repeating group, EMP-RG.



| LINE NO. | SEQ. | SOURCE STATEMENT | |
|----------|------|---------------------------------|-------------------------------------|
| 00001 | | IDENTIFICATION DIVISION, | |
| 00002 | | PROGRAM-ID; EMPL-DEF-1, | |
| 00003 | | DATA DIVISION, | |
| 00004 | | FILE SECTION, | |
| 00005 | | FD EMP-RG, | |
| 00006 | 01 | EMPLOYEE-REC, | |
| 00007 | 02 | EMP-NM | PIC X(21), |
| 00008 | 02 | DEPT-NAME | PIC X(14), |
| 00009 | 02 | COUNT | PIC 9(2), |
| 00010 | 02 | DEP-ENTRY; | |
| 00011 | | OCURS 6 TO 10 TIMES | |
| 00012 | | DEPENDENT ON COUNT | |
| 00013 | | ASCENDING KEY IS DEP-NM, | |
| 00014 | 03 | DEP-NM | PIC X(21), |
| 00015 | 03 | FILLER | PIC X(7), |
| 00016 | 03 | D-SSN | PIC X(9), |
| 00017 | | DEFINITION DIVISION, | |
| 00018 | | DEFINED FILE EMPLOYEES PASSWORD | |
| 00019 | | DEFINED RECORD EMPLOYEE | |
| 00020 | | FROM EMPLOYEE-REC | |
| 00021 | | ALLOW ADD AND DELETE | } Key is EMP-NM |
| 00022 | | IDENTIFIER EMP-NM | |
| 00023 | | ITEM DEPT-NM FROM DEPT-NAME | |
| 00024 | | DEFINED RECORD DEPENDENT | |
| 00025 | | FROM REPEATING GROUP DEP-ENTRY | } Key is EMP-NM and DEP-NM |
| 00026 | | PARENT IS EMPLOYEE | |
| 00027 | | IDENTIFIER DEP-NM | |
| 00028 | | ITEM D-SSN | |

Figure 4-9. Data Definition for Defined File EMPLOYEES, Derived from a Repeating Group

Parent-Child Defined Records Using Two Indexed Files as Sources

Source files

Figure 4-10c shows a third data definition for defined file EMPLOYEES. In this example, employee and dependent records come from sources that are in two different indexed files. Employee records come from EMPFILE (Figure 4-10a); dependent records come from EN-DEP (Figure 4-10b). Each employee record contributes a pointer (item ENTRY) that locates the set of dependent records that are its child records.

File relationships

Figure 4-10 shows the relationship of the two indexed files to each other and to defined file EMPLOYEES.

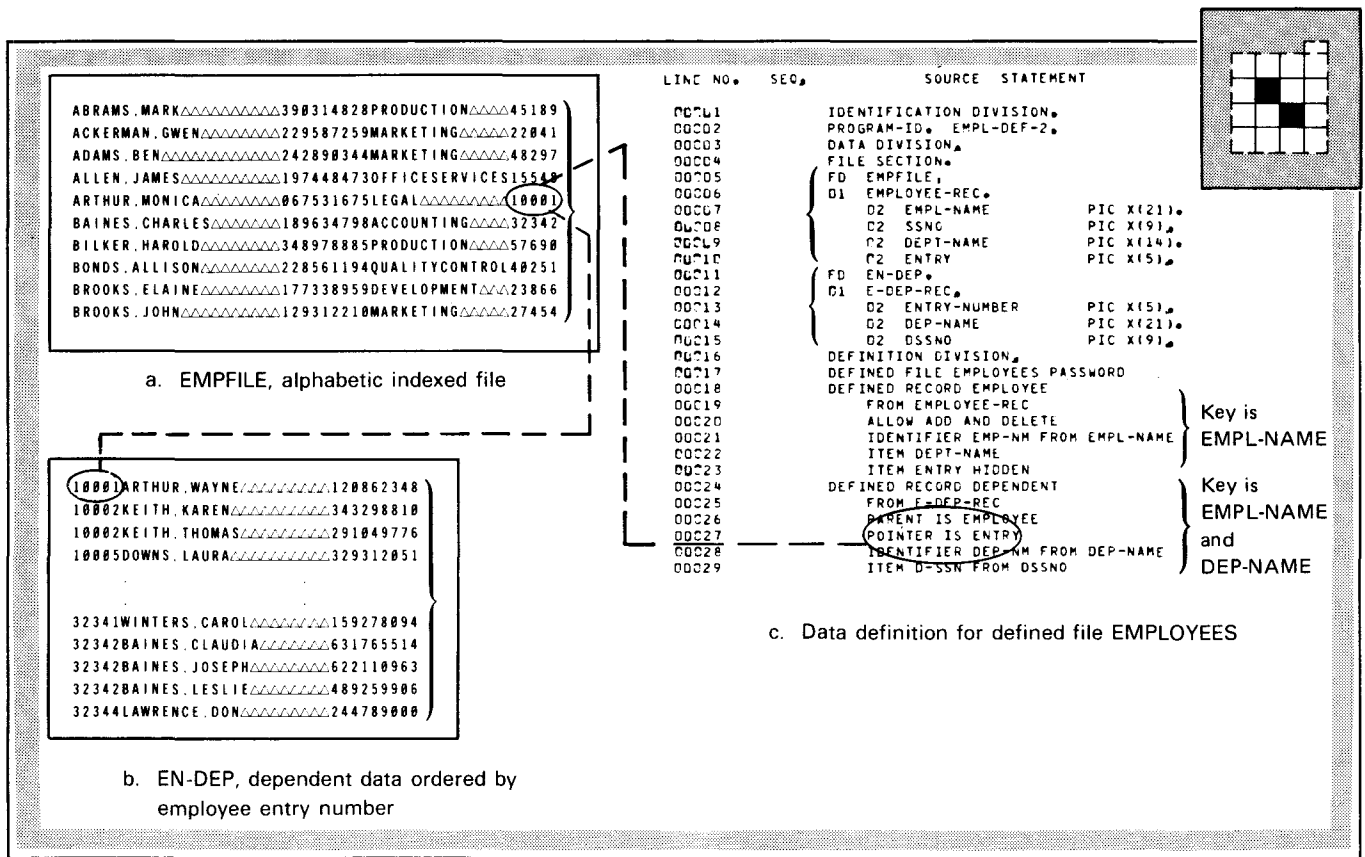
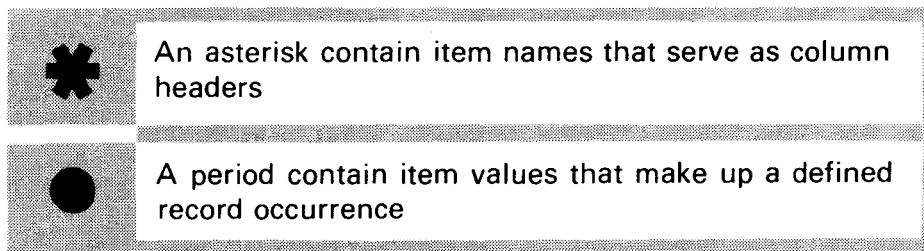


Figure 4-10. Derivation of Defined File EMPLOYEES from Two Distinct Files Using Pointers

EXAMPLES OF HIERARCHICAL DEFINED FILES

Defined File Resulting from Different File Sources*Records delivered
to UNIQUE*

Figure 4-11 shows, for all three examples, the terminal display of EMPLOYEES in response to a UNIQUE LIST command. Lines beginning with:

*Identifiers
and UNIQUE*

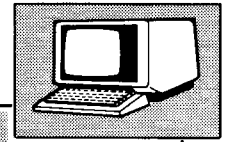
Each dependent record's identifier consists of both employee name (its parent record's identifier) and dependent name, but UNIQUE replaces the employee name with a hyphen to conserve screen space.

*Records delivered
to action programs*

Figure 4-12 shows the first few defined records of defined file EMPLOYEES that IMS delivers to action programs. In response to each GET function call, IMS moves all defined record fields, plus one status byte per field, to the action program.

*Action program record
areas*

Figure 4-13 shows the COBOL, BAL, and RPG II action program descriptions of the record areas receiving parent (EMPLOYEE) and child (DEPENDENT) defined records.



| | |
|--------------------|----------------|
| * EMP-NM | DEPT-NAME |
| * -DEP-NM | D-SSN |
| . ABRAMS, MARK | PRODUCTION |
| . -ABRAMS, JOAN | 326090119 |
| . ACKERMAN, GWEN | MARKETING |
| . ADAMS, BEN | MARKETING |
| . -ADAMS, BONNIE | 298345429 |
| . -ADAMS, SARA | 395006297 |
| . ALLEN, JAMES | OFFICESERVICES |
| . ARTHUR, MONICA | LEGAL |
| . -ARTHUR, WAYNE | 120862348 |
| . BAINES, CHARLES | ACCOUNTING |
| . -BAINES, CLAUDIA | 631765514 |
| . -BAINES, JOSEPH | 622110963 |
| . -BAINES, LESLIE | 489259906 |
| . BILKER, HAROLD | PRODUCTION |

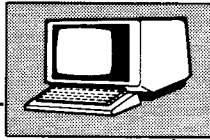
Figure 4-11. First Few EMPLOYEES Records, as Listed at a Terminal by UNIQUE

| 8 | 12 | 1 | 10 | 16 |
|----|----------------------|---------|-------|--------------|
| 01 | WORK-AREA. | WORK | DSECT | WORK AR |
| 02 | EMPLOYEE. | EMPL | EQU | * |
| 03 | EMPLOYEE. | EEMP | DS | CL21 EMPLOYE |
| 04 | EMP-NM PIC X(21). | EDEPT | DS | CL14 EMPLOYE |
| 04 | DEPT-NAME PIC X(14). | EEMP#S | DS | C EMPLOYE |
| 03 | S-EMPLOYEE. | EDEPT#S | DS | C EMPLOYE |
| 04 | S-EMP-NM PIC X. | DEP | EQU | * |
| 04 | S-DEPT-NAME PIC X. | EDEMP | DS | CL21 DEP-EMP |
| 03 | DEPENDENT. | EDDEP | DS | CL21 DEP-DEP |
| 04 | EMP-NM PIC X(21). | EDSS | DS | CL9 DEP-DEP |
| 04 | DEP-NM PIC X(21). | EDEMP#S | DS | C DEP-EMP |
| 04 | D-SSN PIC X(09). | EDDEP#S | DS | C DEP-DEP |
| 03 | S-DEPENDENT. | EDSS#S | DS | C DEP-DEP |
| 04 | S-EMP-NM PIC X. | | | |
| 04 | S-DEP-NM PIC X. | | | |
| 04 | S-D-SSN PIC X. | | | |

a. Description of EMPLOYEE and DEPENDENT in COBOL action program

b. Description of EMPLOYEE a

Figure 4-13. Action Program Descriptions



```

PT - NAME
- SSN
DDUCTION
26090119
RKETING
RKETING
98345429
95006297
FICESERVICES
SAL
20862348
COUNTING
31765514
22110963
39259906
DDUCTION
    
```

```

ABRAMS, MARK^^^^^^^^^^P
ABRAMS, MARK^^^^^^^^^^A
ACKERMAN, GWEN^^^^^^^^^N
ADAMS, BEN^^^^^^^^^^^^^N
ADAMS, BEN^^^^^^^^^^^^^A
ADAMS, BEN^^^^^^^^^^^^^A
ALLEN, JAMES^^^^^^^^^^C
ARTHUR, MONICA^^^^^^^^^I
ARTHUR, MONICA^^^^^^^^^A
BAINES, CHARLES^^^^^^^^^A
BAINES, CHARLES^^^^^^^^^I
BAINES, CHARLES^^^^^^^^^E
BAINES, CHARLES^^^^^^^^^I
BILKER, HAROLD^^^^^^^^^I
    
```

Figure 4-12. First Few EMPLOYEE

as Listed at a Terminal by UNIQUE

| 1 | 10 | 16 | |
|---------|-------|------|--------------------------------------|
| WORK | DSECT | | WORK AREA |
| EMPL | EQU | * | |
| EEMP | DS | CL21 | EMPLOYEE NAME |
| EDEPT | DS | CL14 | EMPLOYEE DEPARTMENT NAME |
| EEMP#S | DS | C | EMPLOYEE NAME STATUS BYTE |
| EDEPT#S | DS | C | EMPLOYEE DEPARTMENT NAME STATUS BYTE |
| DEP | EQU | * | |
| EDEMP | DS | CL21 | DEP-EMPLOYEE NAME |
| EDDEP | DS | CL21 | DEP-DEPENDENT NAME |
| EDSS | DS | CL9 | DEP-DEPENDENT SOC SEC NO |
| EDEMP#S | DS | C | DEP-EMPLOYEE NAME STATUS BYTE |
| EDDEP#S | DS | C | DEP-DEPENDENT NAME STATUS BYTE |
| EDSS#S | DS | C | DEP-DEPENDENT SOC SEC NO STATUS BYTE |

b. Description of EMPLOYEE and DEPENDENT in BAL action program

Figure 4-13. Action Program Descriptions of Defined Records EMPLOYEE and DEPENDENT

EXAMPLES OF HIERARCHICAL DEFINED FILES

| | | |
|-----------------|-----------------|-----------|
| ABRAMS, MARK | PRODUCTION | |
| ABRAMS, MARK | ABRAMS, JOAN | 326090119 |
| ACKERMAN, GWEN | MARKETING | |
| ADAMS, BEN | MARKETING | |
| ADAMS, BEN | ADAMS, BONNIE | 298345429 |
| ADAMS, BEN | ADAMS, SARA | 395006297 |
| ALLEN, JAMES | OFFICESERVICES | |
| ARTHUR, MONICA | LEGAL | |
| ARTHUR, MONICA | ARTHUR, WAYNE | 120862348 |
| BAINES, CHARLES | ACCOUNTING | |
| BAINES, CHARLES | BAINES, CLAUDIA | 631765514 |
| BAINES, CHARLES | BAINES, JOSEPH | 622110963 |
| BAINES, CHARLES | BAINES, LESLIE | 489259906 |
| BILKER, HAROLD | PRODUCTION | |

Figure 4-12. First Few EMPLOYEES Records, as Delivered to an Action Program

NAME
 DEPARTMENT NAME
 NAME STATUS BYTE
 DEPARTMENT NAME STATUS BYTE

EMPLOYEE NAME
 DEPENDENT NAME
 DEPENDENT SOC SEC NO
 EMPLOYEE NAME STATUS BYTE
 DEPENDENT NAME STATUS BYTE
 DEPENDENT SOC SEC NO STATUS BYTE

and DEPENDENT in BAL action program

FILE DESCRIPTION SPECIFICATIONS

| PAGE NO | FORM TYPE | LINE NO | FILE NAME | NOT USED | FILE TYPE | FILE DESIGNATION | END OF FILE | SEQUENCE | FILE FORMAT | BLOCK LENGTH | RECORD LENGTH | FILE PROCESSING MODE | | EXTENSION OR LINE COUNTER CODE | DEVICE | NOT USED | LABELS |
|---------|-----------|---------|-----------|----------|-----------|------------------|-------------|----------|-------------|--------------|---------------|------------------------------------|---------------------|--------------------------------|--------|----------|--------|
| | | | | | | | | | | | | KEY OR RECORD ADDRESS FIELD LENGTH | RECORD ADDRESS TYPE | | | | |
| 1 | F | 0,1 | EMPLOYEE | UC | F | | | | 1112R21A1 | | | APL | ON | DISK | | | S |
| 2 | F | 0,2 | EMPINFD | IP | | | | | | | | | | | | | |

INPUT FORMAT SPECIFICATIONS

| PAGE NO | FORM TYPE | LINE NO | FILE NAME | SEQUENCE | OPTIONAL | RECORD IDENTIFYING INDICATOR OR .. | RECORD IDENTIFICATION CODES | | | | | | | | | FIELD LOCATION | | FIELD DESCRIPTION |
|---------|-----------|---------|-----------|----------|----------|------------------------------------|-----------------------------|-----------|----------|-----------|----------|-----------|------|----|-------------------|----------------|---------|-------------------|
| | | | | | | | POSITION | CHARACTER | POSITION | CHARACTER | POSITION | CHARACTER | FROM | TO | DECIMAL POSITIONS | FIELD NAME | | |
| 1 | F | 0,1 | EMPLOYEE | 01 | | | | | | | | | | | | | | |
| 2 | F | 0,2 | | | | | | | | | | | | 21 | 35 | | EMP-INM | |
| 3 | F | 0,3 | | | | | | | | | | | 22 | 51 | | DEPT-NM | | |
| 4 | F | 0,4 | | | | | | | | | | | | | | | | |
| 5 | F | 0,5 | BB | 02 | | | | | | | | | | | | | | |
| 6 | F | 0,6 | | | | | | | | | | | | 42 | 51 | | DEP-INM | |
| 7 | F | 0,7 | | | | | | | | | | | 43 | 51 | | D-SSN | | |
| 8 | F | 0,8 | | | | | | | | | | | | | | | | |
| 9 | F | 0,9 | EMPINFD | | | | | | | | | | | | | | | |

c. Description of EMPLOYEE and DEPENDENT in RPG II action program

EXAMPLE OF SUPPLEMENTS

4.3. EXAMPLE OF SUPPLEMENTS IN A DEFINED FILE

Preview of example Figures 4-14 through 4-19 show defined file DEPENDENTS, defined record DEP-RECORD, and its supplements.

Source records The data definition for DEPENDENTS in Figure 4-16 shows the use of supplements in a defined file. Each defined record in DEPENDENTS comes from three different records on disk. Two of these records come from indexed file DEPFILE, shown in Figure 4-14. Another record comes from indexed file EMPFILE, shown in Figure 4-15.

Pairs of records The records of indexed file DEPFILE occur in pairs: two records for JOAN ABRAMS, two for BONNIE ADAMS, etc. The first record supplies the primary part of DEP-RECORD; the second supplies a supplement.

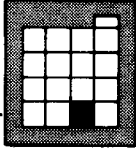
Record keys Both records have keys in the same character positions, 1 through 22. The keys' values differ only in character position 22. The first record contains the space character, while the second contains the number 1, as specified in the FILL KEY TO '1' clause in SUPPLEMENT DEP-PART-1. These values put the records in ascending order and identify the record type as either DEPENDENT-REC or DEPENDENT-REC-TRAILER.

IMS using record type IMS adds, deletes, and displays both types of records together. It uses the record type to act upon the records as a pair. If the first record is missing, IMS ignores the second. If the second record is missing, IMS supplies spaces for item EMPLOYEE. For instance, IMS ignores employee records for GWEN ACKERMAN and JAMES ALLEN (see Figure 4-15) because they have no corresponding dependent records.

Accessing records with a pointer The second file, EMPFILE, contains only one type of record, EMPLOYEE-REC, which supplements defined record DEP-RECORD. You access EMPLOYEE-REC with a pointer. As a record in an indexed file (EMPFILE), EMPLOYEE-REC contains a record key. IMS builds a pointer from NAME-EMPL in record DEPENDENT-REC-TRAILER. It then searches record EMPLOYEE-REC to:

- match the pointer against record key EMPL-NAME; and
- locate the secondary part of the record.

If it does not find an EMPLOYEE-REC record, IMS supplies zeros for item SSNO in the defined record supplement.

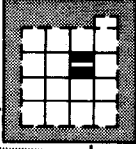


```

ABRAMS, MARK△△△△△△△△△△△△△△△△390314828PRODUCTION△△△△△45189
ACKERMAN, GWEN△△△△△△△△△△△△△△△△229587259MARKETING△△△△△22041
ADAMS, BEN△△△△△△△△△△△△△△△△242890344MARKETING△△△△△48297
ALLEN, JAMES△△△△△△△△△△△△△△△△197448473OFFICESERVICES15548
ARTHUR, MONICA△△△△△△△△△△△△△△△△067531675LEGAL△△△△△△△△△10001
BAINES, CHARLES△△△△△△△△△△△△△△△△189634798ACCOUNTING△△△△△32342
BILKER, HAROLD△△△△△△△△△△△△△△△△348978885PRODUCTION△△△△△57690
BONDS, ALLISON△△△△△△△△△△△△△△△△228561194QUALITYCONTROL40251
BROOKS, ELAINE△△△△△△△△△△△△△△△△177338959DEVELOPMENT△△△23866
BROOKS, JOHN△△△△△△△△△△△△△△△△129312210MARKETING△△△△△27454

```

Figure 4-15. Excerpt from EMPFILE, an Indexed Employee File

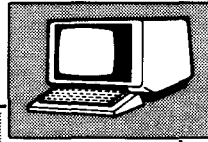


```

LINE NO.  SEQ.  SOURCE STATEMENT
00C01      IDENTIFICATION DIVISION
00C02      PROGRAM-ID, SEC-PART-DEF
00C03      DATA DIVISION
00C04      FILE SECTION
00C05      FD EMPFILE
00C06      01 DEPENDENT-REC
00C07      02 DEP-NAME          PIC X(21)
00C08      02 RCD-TYPE        PIC X
00C09      02 DSSNO          PIC X(9)
00C10      02 FILLER          PIC X(12)
00C11      01 DEPENDENT-REC-TRAILER
00C12      02 DEP-ID         PIC X(21)
00C13      02 TYPE-RCD       PIC X
00C14      02 NAME-EMPL     PIC X(21)
00C15      FD EMPFILE
00C16      01 EMPLOYEE-REC
00C17      02 EMPL-NAME      PIC X(21)
00C18      02 SSNO          PIC X(9)
00C19      02 DEPT-NAME     PIC X(14)
00C20      02 ENTRY         PIC X(5)
00C21      DEFINITION DIVISION
00C22      DEFINED FILE DEPENDENTS
00C23      DEFINED RECORD DEP-RECORD
00C24      FROM DEPENDENT-REC
00C25      ALLOW ADD AND DELETE
00C26      IDENTIFIER DNAME FROM DEP-NAME
00C27      ITEM DSSNO
00C28      SUPPLEMENT DEP-PART-1
00C29      FROM DEPENDENT-REC-TRAILER
00C30      FILL KEY TO '1' ASSUMES CONTROLLED ROLE IN UPDATE
00C31      ITEM EMPLOYEE FROM NAME-EMPL
00C32      SUPPLEMENT DEP-PART-2
00C33      FROM EMPLOYEE-REC
00C34      POINTER IS EMPLOYEE
00C35      ITEM SSNO

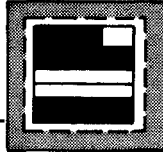
```

Figure 4-16. Data Definition for Defined File DEPENDENTS Showing Supplements



| DNAME | DSSNO | EMPLOYEE | SSNO |
|---------------|-----------|----------------|-----------|
| ABRAMS, JOAN | 326090119 | ABRAMS, MARK | 309314828 |
| ADAMS, BONNIE | 298345429 | ADAMS, BEN | 242890344 |
| ADAMS, SARA | 395006297 | ADAMS, BEN | 242890344 |
| ARTHUR, WAYNE | 120862348 | ARTHUR, MONICA | 067531675 |

Figure 4-17. First Few DEPENDENTS Records, as Listed at a Terminal by UNIQUE



```

ABRAMS, JOAN^^^^^^^^^^326090119ABRAMS, MARK^^^^^^^^^^309314828
ADAMS, BONNIE^^^^^^^^^^298345429ADAMS, BEN^^^^^^^^^^242890344
ADAMS, SARA^^^^^^^^^^395006297ADAMS, BEN^^^^^^^^^^242890344
ARTHUR, WAYNE^^^^^^^^^^120862348ARTHUR, MONICA^^^^^^^^^^067531675
    
```

Figure 4-18. First Few DEPENDENTS Records, as Delivered to an Action Program

EXAMPLE OF SUPPLEMENTS

```

8 12
01 WORK AREA.
  02 DEPENDE.
    03 DEP-RECORD.
      04 DNAME          PIC X(21).
      04 DSSNO          PIC X(09).
      04 EMPLOYEE      PIC X(21).
      04 SSNO           PIC X(09).
    03 S-DEP-RECORD.
      04 S-DNAME        PIC X.
      04 S-DSSNO        PIC X.
      04 S-EMPLOYEE     PIC X.
      04 S-SSNO         PIC X.
  
```

a. Description of DEP-RECORD in COBOL action program

```

1 10 16
WORK DSECT WORK AREA
DP-REC EQU *
EDPNM DS CL21 DEPENDENT NAME
EDPSSN DS CL9 DEPENDENT SOC SEC NO
EEMPL DS CL21 EMPLOYEE NAME
ESSNO DS CL9 EMPLOYEE SOC SEC NO
EDPNM#S DS C DEPENDENT NAME STATUS BYTE
EDPSSN#S DS C DEPENDENT SOC SEC NO STATUS BYTE
EEMPL#S DS C EMPLOYEE NAME STATUS BYTE
ESSNO#S DS C EMPLOYEE SOC SEC NO STATUS BYTE
  
```

b. Description of DEP-RECORD (labeled DP-REC) in BAL action program

FILE DESCRIPTION SPECIFICATIONS

| PAGE NO | LINE NO | FILE NAME | FILE TYPE | FILE DESIGNATION | SEQUENCE | FILE FORMAT | FILE PROCESSING MODE | | EXTENSION/FILE COUNTER | LABELS |
|---------|---------|-----------|-----------|------------------|----------|-------------|-----------------------------|---------------------------|------------------------|--------|
| | | | | | | | KEY FIELD STARTING POSITION | KEY FIELD ENDING POSITION | | |
| 01 | 1 | DEPS11 | UC | F | 1 | 86R21AI | | | DISK | S |
| 01 | 2 | EMPIINF | IP | | | | | | | |

INPUT FORMAT SPECIFICATIONS

| PAGE NO | LINE NO | FILE NAME | RECORD IDENTIFICATION | | | FIELD LOCATION | | FIELD DESCRIPTION |
|---------|---------|-----------|-----------------------------|----------|-----------|----------------|-------|-------------------|
| | | | RECORD IDENTIFICATION CODES | POSITION | CHARACTER | FROM | TO | |
| 01 | 1 | DEPS11 | AA | 01 | | | | |
| 01 | 2 | | | | | | 21 | DNAME |
| 01 | 3 | | | | | | 22 30 | DSSNO |
| 01 | 4 | | | | | | 31 51 | EMPL |
| 01 | 5 | | | | | | 52 60 | SSNO |
| 01 | 7 | EMPIINF | | | | | | |

c. Description of DEP-RECORD in RPG II action program

Figure 4-19. Action Program Descriptions of Defined Record DEP-RECORD

Intentionally left blank

EXAMPLE OF SUBFILE

4.4. EXAMPLE OF A SUBFILE***Data definition***

Figure 4-20 shows how you can use a subfile definition to restrict access to a defined file. Both this data definition and the one shown in Figure 4-2 use the same source data (from indexed file EMPFILE in Figure 4-1). They both also make defined file EMPLS1 available to action programs and, through UNIQUE, to terminal operators.

Records delivered

As in Figure 4-2, you can access data through defined file name EMPLS1. You can also access subrecord SUB-EMPS, but only through subfile name SUBFIL. Figures 4-21 and 4-22 show that, for SUBFIL, IMS delivers only two items, employee name and social security number, to terminals and action programs. Their item names, used as column headers by UNIQUE, change from EMP-NM to NAME-OF-EMP and from SSNO to SSNUMBER.

Action program record areas

Figure 4-23 shows COBOL, BAL, and RPG II action program descriptions of the record areas receiving SUB-EMPS.

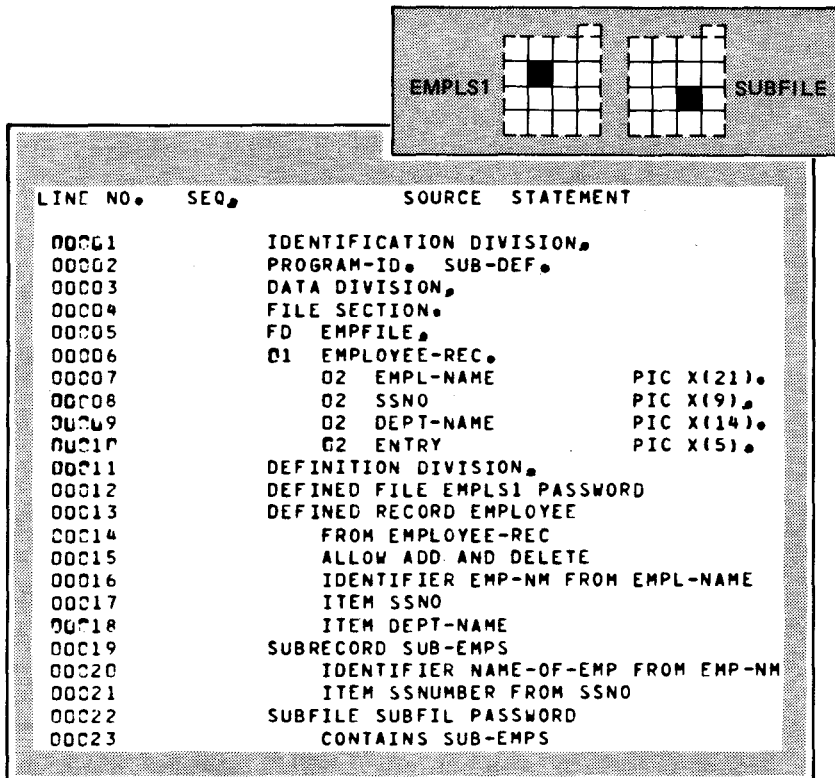


Figure 4-20. Subfile Definition of SUBFIL, Restricting Access to Defined File EMPLS1

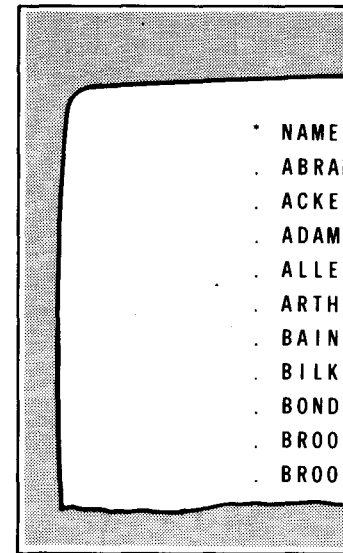


Figure 4-21. First F

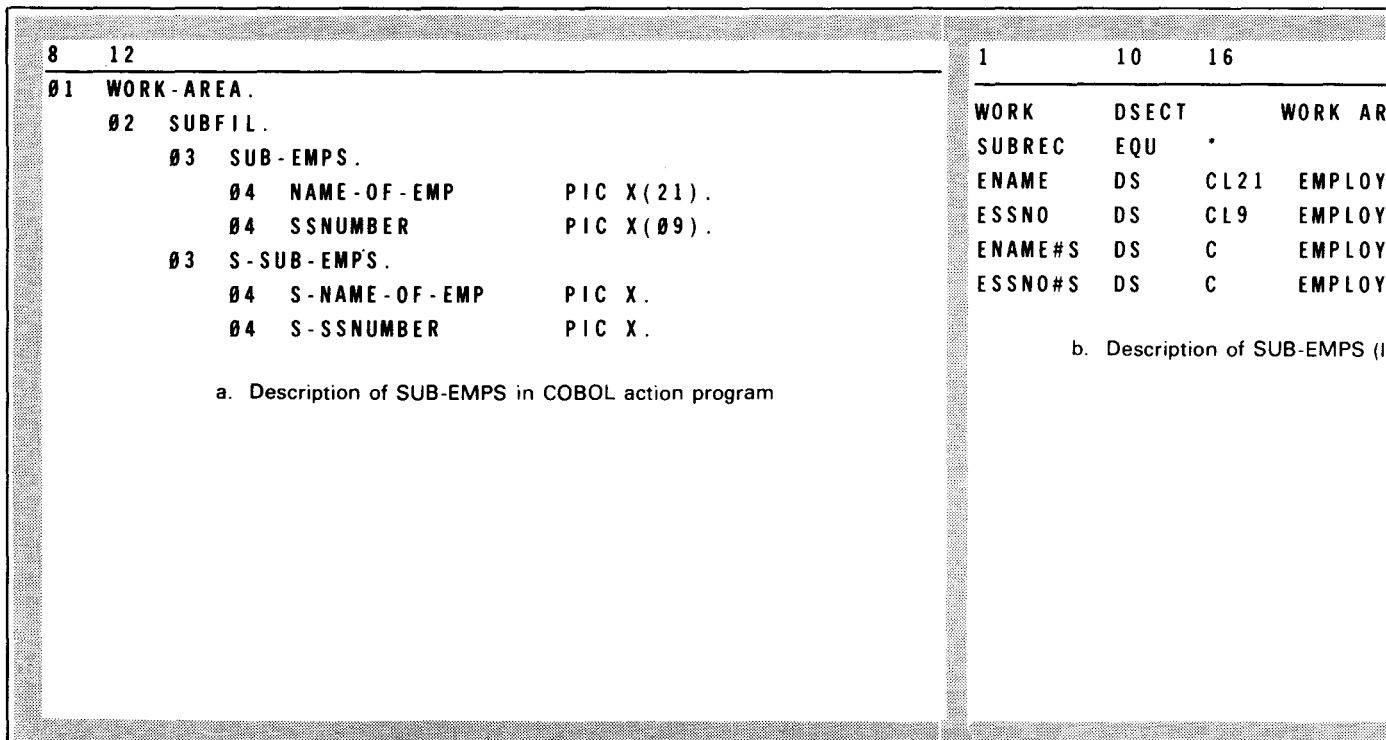
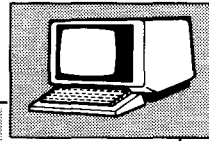


Figure 4-23. Action Program



| * NAME - OF - EMP | SSNUMBER |
|-------------------|-----------|
| ABRAMS, MARK | 309314828 |
| ACKERMAN, GWEN | 229587259 |
| ADAMS, BEN | 242890344 |
| ALLEN, JAMES | 197448473 |
| ARTHUR, MONICA | 067531675 |
| BAINES, CHARLES | 189634798 |
| BILKER, HAROLD | 348978885 |
| BONDS, ALLISON | 228561194 |
| BROOKS, ELAINE | 177338959 |
| BROOKS, JOHN | 129312210 |

| | |
|-----------------|---------------------|
| ABRAMS, MARK | ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲ 30 |
| ACKERMAN, GWEN | ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲ 22 |
| ADAMS, BEN | ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲ 24 |
| ALLEN, JAMES | ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲ 19 |
| ARTHUR, MONICA | ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲ 06 |
| BAINES, CHARLES | ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲ 18 |
| BILKER, HAROLD | ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲ 34 |
| BONDS, ALLISON | ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲ 22 |
| BROOKS, ELAINE | ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲ 17 |
| BROOKS, JOHN | ▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲▲ 12 |

Figure 4-21. First Few SUBFIL Records, as Listed at a Terminal by UNIQUE

Figure 4-22. First Few SUBFIL Records, as Listed at a Terminal by UNIQUE

| 1 | 10 | 16 | |
|---------|-------|-----------|---------------------------------|
| WORK | DSECT | WORK AREA | |
| SUBREC | EQU * | | |
| ENAME | DS | CL21 | EMPLOYEE NAME |
| ESSNO | DS | CL9 | EMPLOYEE SOC SEC NO |
| ENAME#S | DS | C | EMPLOYEE NAME STATUS BYTE |
| ESSNO#S | DS | C | EMPLOYEE SOC SEC NO STATUS BYTE |

| FORM TYPE | | FILE TYPE | FILE DESIGNATION | FILE PROCESSING MODE | EXTENSION OR LINE COUNTER CODE |
|-----------|---|-----------|------------------|----------------------|--------------------------------|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 0,1 | F | SUBFIL | UC | 56R21AI | DISK |
| 0,2 | F | EMPINFD | IP | | |

b. Description of SUB-EMPS (labeled SUB-REC) in BAL action program

| FORM TYPE | | FILE TYPE | SEQUENCE NUMBER | RECORD IDENTIFICATION INDICATOR | RECORD IDENTIFICATION CODES | | | INPUT FORM | | | | | |
|-----------|---|-----------|-----------------|---------------------------------|-----------------------------|---|---|------------|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 0,1 | F | SUBFIL | AA | 01 | | | | | | | | | |
| 0,2 | F | | | | | | | | | | | | |
| 0,3 | F | | | | | | | | | | | | |
| 0,4 | F | | | | | | | | | | | | |
| 0,5 | F | EMPINFD | | | | | | | | | | | |

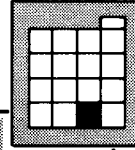
c. Description of SUB-EMPS in RPG II action program

Figure 4-23. Action Program Descriptions of Subrecord SUB-EMPS

EXAMPLE OF INTERRELATED DEFINED FILES

4.5. EXAMPLE OF INTERRELATED DEFINED FILES

- Source files* Figures 4-24 through 4-29 show how you can use multiple defined files and supplements to access data in different ways. Figure 4-24 shows the source records that come from indexed files EMPFILE AND COFILE. Both files contain the same data, but that data is keyed in different ways.
- Defined files* For defined file EMPLOYEES, you receive data alphabetically according to employee name. The file contains defined record EMPLOYEE. For defined file COMPANY, you can get the data according to division name, department name, and employee name. The file contains hierarchical records CO-DIVISION, DEPARTMENT, and EMPLOYEES.
- Defined records and supplements* Defined record EMPLOYEE has the same source (EMPLOYEE-REC) as supplement EMP-SUPP in defined file COMPANY. Supplement EMP-1 has the same source (COMPANY-REC) as defined records CO-DIVISION, DEPARTMENT, and EMPLOYEE in defined file COMPANY. In supplement EMP-1, the POINTER clause has the same effect as an ITEM clause. DIV-NM, DEPT-NM, and EMP-NM (used as identifiers for the records in defined file COMPANY) are carried down to supplement EMP-1, where their data can be changed.
- Changing the records* According to the data definitions, you can add, delete, or make changes to EMPLOYEE, but these changes do not affect the data you will get for supplement EMP-SUPP. You cannot add, delete, or make changes to any of the defined records or supplements in defined file COMPANY. COMPANY is used for retrieval purposes only.
- Data definitions* Figure 4-25 shows the data definitions for defined files EMPLOYEE and COMPANY.
- Records delivered* Figures 4-26 through 4-29 show how UNIQUE lists the first few EMPLOYEES and COMPANY records at a terminal and how IMS delivers them to action programs.



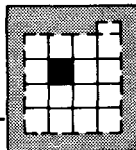
ABRAMS, MARK△△△△△△△△△△DIVISION01PRODUCTION△△△△45189△△2200012
ACKERMAN, GWEN△△△△△△△△△△DIVISION03MARKETING△△△△△22041△△3250016
ADAMS, BEN△△△△△△△△△△DIVISION03MARKETING△△△△△48297△△3810017
ALLEN, JAMES△△△△△△△△△△DIVISION09OFFICESERVICES15548△△1950012
ARTHUR, MONICA△△△△△△△△△△DIVISION05LEGAL△△△△△10001△△4950019
BAINES, CHARLES△△△△△△△△△△DIVISION07ACCOUNTING△△△△△32342△△3700012
BILKER, HAROLD△△△△△△△△△△DIVISION01PRODUCTION△△△△△57690△△2000011
BONDS, ALLISON△△△△△△△△△△DIVISION11QUALITYCONTROL40251△△2750014
BROOKS, ELAINE△△△△△△△△△△DIVISION08DEVELOPMENT△△△23866△△3000016
BROOKS, JOHN△△△△△△△△△△DIVISION03MARKETING△△△△△27454△△4000018

a. Excerpt from EMPFILE, an indexed employee file

DIVISION01PRODUCTION△△△△ABRAMS, MARK△△△△△△△△△△45189△△2200012
DIVISION01PRODUCTION△△△△BILKER, HAROLD△△△△△△△△△△57690△△2000011
DIVISION01PRODUCTION△△△△BUNDY, ELLEN△△△△△△△△△△16987△△2100012
DIVISION01PRODUCTION△△△△CANE, GERALD△△△△△△△△△△39054△△2400013
DIVISION01PRODUCTION△△△△CARSON, MELANIE△△△△△△△△△△20931△△2150012
DIVISION01PRODUCTION△△△△CONNELL, THEODORE△△△△△11956△△2350012
DIVISION01PRODUCTION△△△△CUMMINGS, DONALD△△△△△34710△△2000012
DIVISION01PRODUCTION△△△△DAVIS, JENNIFER△△△△△51246△△2050012
DIVISION01PRODUCTION△△△△DAWSON, CYNTHIA△△△△△43897△△2350013
DIVISION01PRODUCTION△△△△DELANEY, SANDRA△△△△△39011△△2200012

b. Excerpt from COFILE, an indexed company file

Figure 4-24. Indexed Files EMPFILE and COFILE



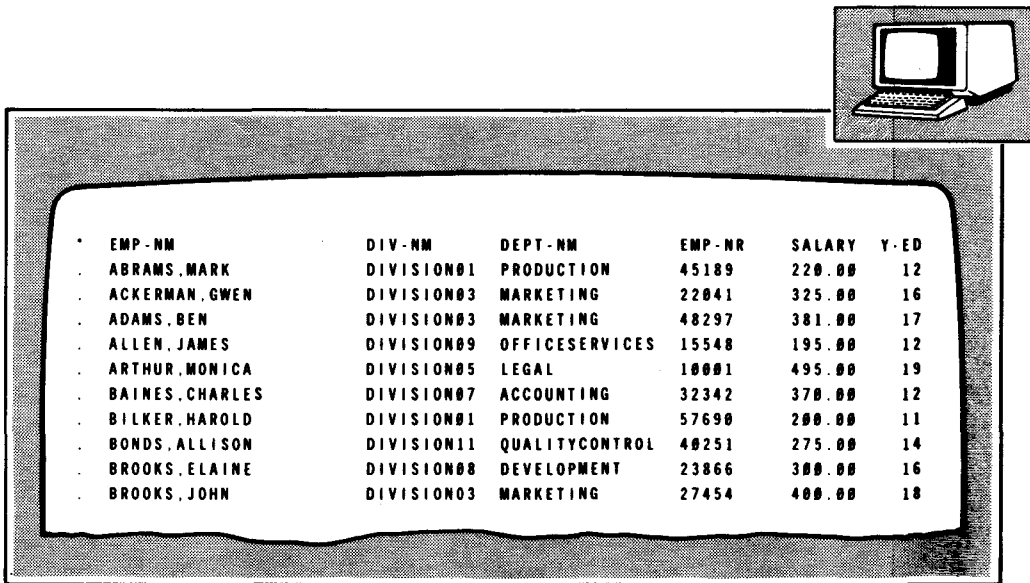
```
LINE NO.  SEQ.  SOURCE STATEMENT
00001      IDENTIFICATION DIVISION.
00002      PROGRAM-ID.  MULT-DATA-DEF1.
00003      DATA DIVISION.
00004      FILE SECTION.
00005      FD EMPFILE.
00006      01 EMPLOYEE-REC.
00007         02 EMPL-NAME           PIC X(21).
00008         02 DIVISION-NAME      PIC X(10).
00009         02 DEPT-NAME          PIC X(14).
00010         02 EMP-NR             PIC X(5).
00011         02 SALARY-AMT         PIC 9(5)V99.
00012         02 YRS-EDUC          PIC 9(2).
00013      FD COFILE.
00014      01 COMPANY-REC.
00015         02 DIVISION-NAME      PIC X(10).
00016         02 DEPT-NAME          PIC X(14).
00017         02 EMPL-NAME          PIC X(21).
00018      DEFINITION DIVISION.
00019      DEFINED FILE EMPLOYEES PASSWORD
00020      DEFINED RECORD EMPLOYEE
00021      FROM EMPLOYEE-REC
00022      ALLOW ADD AND DELETE
00023      IDENTIFIER EMP-NM FROM EMPL-NAME
00024      ITEM DIV-NM FROM DIVISION-NAME ALLOW CHANGE MUST ADD
00025      ITEM DEPT-NM FROM DEPT-NAME ALLOW CHANGE MUST ADD
00026      ITEM EMP-NR MUST ADD
00027      ITEM SALARY FROM SALARY-AMT ALLOW CHANGE
00028      ITEM Y-ED FROM YRS-EDUC
00029      SUPPLEMENT EMP-1
00030      FROM COMPANY-REC
00031      POINTER IS DIV-NM, DEPT-NM, EMP-NM
00032      ASSUMES CONTROLLED
```

a. Defined file EMPLOYEES

```
LINE NO.  SEQ.  SOURCE STATEMENT
00001      IDENTIFICATION DIVISION.
00002      PROGRAM-ID.  MULT-DATA-DEF2.
00003      DATA DIVISION.
00004      FILE SECTION.
00005      FD COFILE.
00006      01 COMPANY-REC.
00007         02 DIVISION-NAME      PIC X(10).
00008         02 DEPT-NAME          PIC X(14).
00009         02 EMPL-NAME          PIC X(21).
00010      FD EMPFILE.
00011      01 EMPLOYEE-REC.
00012         02 EMPL-NAME           PIC X(21).
00013         02 DIVISION-NAME      PIC X(10).
00014         02 DEPT-NAME          PIC X(14).
00015         02 EMP-NR             PIC X(5).
00016         02 SALARY-AMT         PIC 9(5)V99.
00017         02 YRS-EDUC          PIC 9(2).
00018      DEFINITION DIVISION.
00019      DEFINED FILE COMPANY
00020      DEFINED RECORD CO-DIVISION
00021      FROM CONTROL BREAK IN COMPANY-REC
00022      IDENTIFIER DIV-NM FROM DIVISION-NAME
00023      DEFINED RECORD DEPARTMENT
00024      FROM CONTROL BREAK IN COMPANY-REC
00025      PARENT IS CO-DIVISION
00026      IDENTIFIER DEPT-NM FROM DEPT-NAME
00027      DEFINED RECORD EMPLOYEE
00028      FROM COMPANY-REC
00029      PARENT IS DEPARTMENT
00030      IDENTIFIER EMP-NM FROM EMPL-NAME
00031      SUPPLEMENT EMP-SUPP
00032      FROM EMPLOYEE-REC
00033      POINTER IS EMP-NM
00034      ASSUMES NEUTRAL
00035      ITEM EMP-NR
00036      ITEM SALARY FROM SALARY-AMT
00037      ITEM Y-ED FROM YRS-EDUC
```

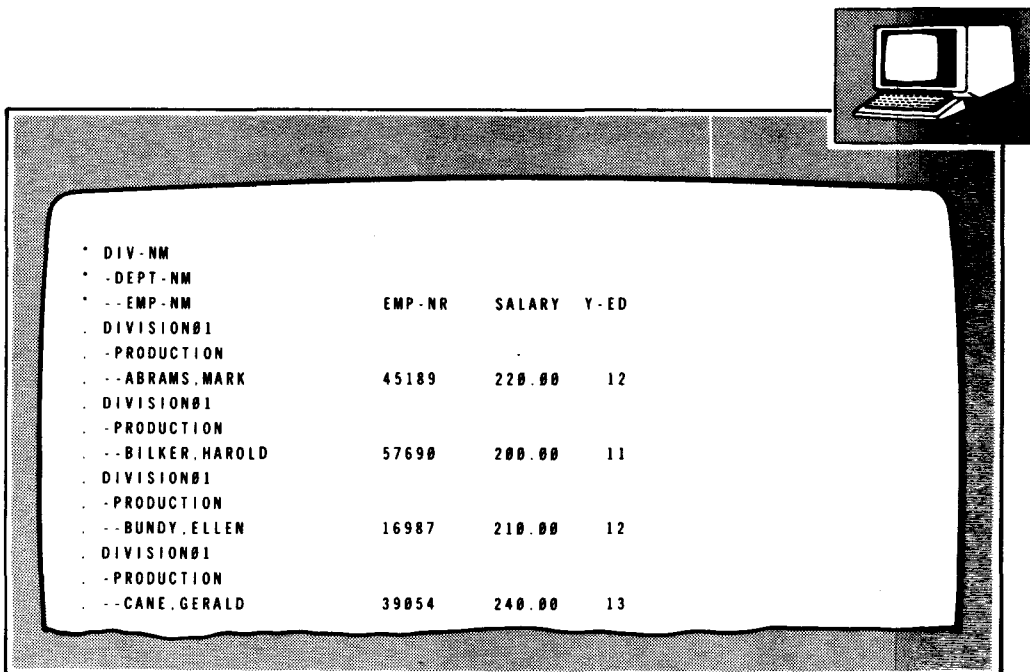
b. Defined file COMPANY

Figure 4-25. Data Definitions for Related Defined Files EMPLOYEES and COMPAN



| EMP-NM | DIV-NM | DEPT-NM | EMP-NR | SALARY | Y-ED |
|-----------------|------------|----------------|--------|--------|------|
| ABRAMS, MARK | DIVISION01 | PRODUCTION | 45189 | 220.00 | 12 |
| ACKERMAN, GWEN | DIVISION03 | MARKETING | 22041 | 325.00 | 16 |
| ADAMS, BEN | DIVISION03 | MARKETING | 48297 | 381.00 | 17 |
| ALLEN, JAMES | DIVISION09 | OFFICESERVICES | 15548 | 195.00 | 12 |
| ARTHUR, MONICA | DIVISION05 | LEGAL | 10001 | 495.00 | 19 |
| BAINES, CHARLES | DIVISION07 | ACCOUNTING | 32342 | 370.00 | 12 |
| BILKER, HAROLD | DIVISION01 | PRODUCTION | 57690 | 200.00 | 11 |
| BONDS, ALLISON | DIVISION11 | QUALITYCONTROL | 40251 | 275.00 | 14 |
| BROOKS, ELAINE | DIVISION08 | DEVELOPMENT | 23866 | 300.00 | 16 |
| BROOKS, JOHN | DIVISION03 | MARKETING | 27454 | 400.00 | 18 |

Figure 4-26. First Few EMPLOYEES Records, as Listed at a Terminal by UNIQUE



| DIV-NM | DEPT-NM | EMP-NM | EMP-NR | SALARY | Y-ED |
|------------|------------|----------------|--------|--------|------|
| DIVISION01 | PRODUCTION | ABRAMS, MARK | 45189 | 220.00 | 12 |
| DIVISION01 | PRODUCTION | BILKER, HAROLD | 57690 | 200.00 | 11 |
| DIVISION01 | PRODUCTION | BUNDY, ELLEN | 16987 | 210.00 | 12 |
| DIVISION01 | PRODUCTION | CANE, GERALD | 39054 | 240.00 | 13 |

Figure 4-27. First Few COMPANY Records, as Listed at a Terminal by UNIQUE

5. The Data Definition Processor

5.1. EXECUTING THE DATA DEFINITION PROCESSOR

Characteristics

After writing your data definition, you must submit it to the data definition processor. The processor, whose module name is DT3DF, is an IMS utility program that:

- creates a data definition record in the named record (NAMEREC) file; and
- produces a printed description of the defined file (5.4) and a diagnostic listing (5.5).

Figure 5-1 shows this process.

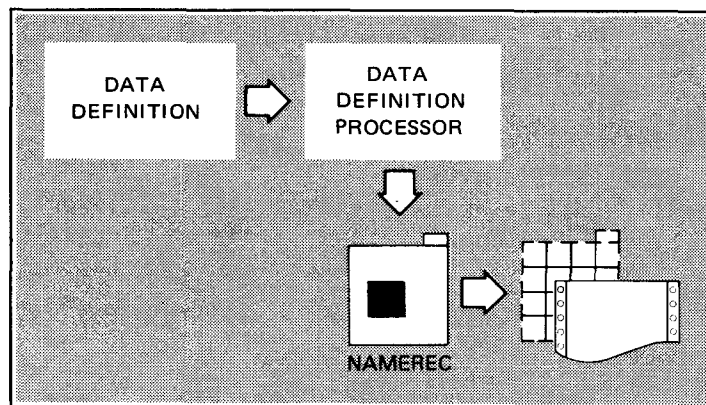


Figure 5-1. Data Definition Processing

EXECUTING THE DATA DEFINITION PROCESSOR

Storing multiple defined files

You can store multiple defined files in the same NAMEREC file, but:

- you can create these multiple files only through separate runs of the data definition processor; and
- you cannot execute the data definition processor while IMS is accessing NAMEREC.

Initializing NAMEREC

You must initialize the NAMEREC file before executing the data definition processor for the first time. You can initialize the NAMEREC file with the NAMEREC file utility or as part of the configuration process. Both are described in the current version of the IMS system support functions user guide, UP-8364. Any time you reinitialize the NAMEREC file, you must recompile all data definitions.

Job control stream elements

The job control stream to execute the data definition processor consists of:

- PARAM statements (5.2);
- other job control statements (5.3); and
- source statements (3.1 through 3.55).

5.2. DATA DEFINITION PROCESSOR OPTIONS

To use specific data definition options, you can present parameters to the data definition processor through the PARAM job control statement. Its format is:

PARAM statement format

```
// PARAM parameters
```

PARAM statement errors

You put PARAM statements directly after the EXEC job control statement (// EXEC DT3DF) in the execution job control stream. The data definition processor prints these statements on the first page of the diagnostic listing. If there is a PARAM statement format error or an illegal parameter:

- the system console receives a message; and
- the data definition run terminates.

List Options

Format

The format is:

```
// PARAM LST=(spec-1, . . . , spec-n)
```

You can substitute one or more of the following options for spec-1, . . . , spec-n:

Resolving references**A**

Activates the ambiguity mode of resolving references. After the entry is resolved, rather than ending with the last entry in the division, the search for duplicate references in the data definition continues through the other divisions.

Sequence checking**K**

Inhibits source item sequence number checking (in columns 1 through 6). This prevents abnormal termination of the data definition processor due to out-of-sequence statements.

Single-spaced listing**L**

Produces a single-spaced diagnostic and source listing. If you omit this parameter, you get a double-spaced listing. If you do not request a source program listing, the processor produces a single-spaced diagnostic listing only.

Katakana characters**Q**

Allows Katakana character set for defined file, subfile, defined record, identifier, and item names. For more details on the Katakana character set, see the IMS system support functions user guide, UP-8364 (current version).

Source listing**S**

Produces source program listing. If you supply no PARAM statements, this is the default.

EXECUTING THE DATA DEFINITION PROCESSOR

Source and Copy Library Input Options***Naming a source file***

When your data definition source statements are stored in a disk or diskette file, you name the source file by specifying the PARAM statement:

Format // PARAM IN=program-name/file-name

Program-name

Program-name is a 1- to 8-character name for your source data definition program. This is the name on the PROGRAM-ID statement in the identification division.

File-name

File-name is a 1- to 8-character name identifying the file that contains your source data definition program. You must include this name in an LFD job control statement. (See Figure 5-2b.) When you omit file-name, the data definition processor assumes your source program is stored in the system source file, \$Y\$SRC. You don't need a device assignment set when the source program is filed in \$Y\$SRC.

Example

```
// PARAM IN=PAYROLL1/PAYFILE
```

Allowing copy library input

You can allow copy library input by specifying the PARAM statement:

```
// PARAM LIN=file-name
```

File-name

File-name is a 1- to 8-character name identifying your COPY library. You must include this name in an LFD job control statement. (See Figure 5-2.) When you omit file-name, the data definition processor assumes your copy library is named COPY\$. You supply the COPY element-name in your source data definition program through the COPY clause in the data division. (See Figure 3-2.)

Example

```
// PARAM LIN=PAYFILE
```

5.3. EXECUTION RUN STREAMS***Sample job control streams***

Figure 5-2 shows two sample job control streams for executing the data definition processor:


```

// JOB DATADF,.C000
// DVC 20 // LFD PRNTR
// OPTION DUMP
// DVC 50 // VOL DS9999 // LBL NAMEREC // LFD ISAMNRF
// WORK1
// WORK2
// WORK3
// EXEC DT3DF
/$
    source cards
    .
    .
    .
    source cards
/*
/&
// FIN

```

a. Input entered from job control stream

```

// JOB DATADF,.C000
// DVC 20 // LFD PRNTR
// OPTION DUMP
// DVC 50 // VOL DS9999 // LBL NAMEREC // LFD ISAMNRF
// DVC 50 // VOL DS9999 // LBL IMSSRC // LFD PAYFILE
// DVC 50 // VOL DS9999 // LBL COPYLIB // LFD COPYLIB
// WORK1
// WORK2
// WORK3
// EXEC DT3DF
// PARAM IN=PAYROLL1/PAYFILE,LIN=COPYLIB,LST=(L.S)
/$
/&
// FIN

```

b. Input entered from source file

Figure 5-2. Sample Job Control Streams to Execute the Data Definition Processor

EXECUTING THE DATA DEFINITION PROCESSOR

Coding requirements

The data definition processor requires C000 hexadecimal bytes (50K decimal bytes) in main storage. Always include a device assignment set for the NAMEREC file and specify ISAMNRF as the file name on the LFD statement. You must also assign three work files.

Job control stream input

In Figure 5-2a:

- source statements are embedded in the job control stream;
- a double-spaced diagnostic and source listing is assumed; and
- there is no copy library input.

Source file input

In Figure 5-2b:

- the data definition source program, PAYROLL1, is stored in PAYFILE (IN parameter);
- a single-spaced listing is requested (LST parameter); and
- a copy element used in the source program is stored in COPYLIB (LIN parameter).

Coding rules

If you use \$Y\$SRC as the file name on the // PARAM IN statement, you do not need to include the DVC, VOL, LBL or LFD job control statements for a source file. If the file name for your copy library is COPY\$, you can omit the LIN parameter, but you still need a device assignment set with COPY\$ as the file name on the LFD statement.

5.4. DATA DEFINITION OUTPUT LISTING

Contents

The data definition processor produces a printed output listing that contains:

- a listing of the source input (your data definition);
- a COBOL description of the defined file when it successfully creates a data definition record; and
- diagnostic messages (5.5) when it detects errors and cannot create a data definition record.

Example source listings

Source input listings for example data definitions appear in Section 4.

Sample COBOL description

Figure 5-3 shows the COBOL description of defined file DEPENDENTS, described in Figure 4-16. To match a COBOL action program accessing the defined file, the processor describes:

- each defined record as a COBOL group item; and
- one item status byte for each elementary item defined.

Item status bytes

The processor generates each item status byte data name by prefixing the data name of the corresponding elementary item with 'S-'. During online processing, IMS tests for the completeness and validity of data transfer after retrieving a record. IMS uses the item status byte data name to access each item's status byte. The processor changes the level numbers in the COBOL description because it provides an I/O area for the defined file.

Compilation information

The last two lines of output contain the statement DATA DEFINITION COMPLETE, compilation time figures, and the statement SUCCESSFUL COMPILATION or UNSUCCESSFUL COMPILATION.

DATA DEFINITION PROCESSOR OUTPUT

```

THE FOLLOWING IS THE COBOL DESCRIPTION OF THE DEFINED FILE DEPENDE.
U2 DEPENDE.
* DEFINED RECORD
U3 DEP-RECORD
  D4 DNAME PIC X(0U21).
  D4 DSSNO PIC X(0U19).
  D4 EMPLOYEE PIC X(0U21).
  D4 SSNO PIC X(0U09).
* THE DEFINED RECORD WILL AUTOMATICALLY INCLUDE ONE STATUS BYTE FOR EACH ELEMENTARY ITEM DEFINED ABOVE.
U3 S-DEP-RECORD.
  UN S-DNAME PIC X.
  UN S-DSSNO PIC X.
  UN S-EMPLOYEE PIC X.
  UN S-SSNO PIC X.

DD1667 NAMEREC
SECPAR DATA DEFINITION COMPLETE START 19:49:42 END 19:56:09
UPSI SETTING = 00 SUCCESSFUL COMPILATION
    
```

Figure 5-3. COBOL Description of Defined File DEPENDENTS

5.5. ERROR PROCESSING BY THE DATA DEFINITION PROCESSOR

Error checking

When processing your input, the data definition processor checks for syntactical errors and issues diagnostics. If it finds any errors, the processor does not create a data definition record.

Data division rules

For the data division of your input, the processor:

- applies extended COBOL rules;
- applies the COBOL reserved word list; and
- issues COBOL diagnostics.

Definition division rules

For the definition division, the processor applies both standard COBOL rules and its own rules. (See Appendixes A and B.) When it finds any rule violations, it issues its own diagnostics. See Appendix C for a listing of these diagnostics.

Diagnostic message contents

Each diagnostic message contains, in this order:

| | |
|----------|---|
| 1 | the processor-generated line number that has the error; |
| 2 | the diagnostic severity code; |
| 3 | the diagnostic number; and |
| 4 | the diagnostic message text. |

Severity codes

Diagnostic severity definitions are:

| | |
|----------|--|
| C | Changed Issued when you omit or incorrectly use a character, word, clause, entry, or statement in your source program. The processor ignores the statement and continues analyzing the remainder of your source program. It does not create a data definition record. |
| U | Uncorrectable Issued when the processor detects a source language error that causes it to delete a character, word, clause, entry, or statement from the source program. Compilation continues, but other errors can result because of the deleted item. The processor does not create a data definition record. |

Sample outputs

Figure 5-4 shows the output from several unsuccessful runs of the data definition processor.

Abnormal termination

If the data definition processor terminates abnormally and issues no diagnostic messages, the problem with your data definition could possibly be:

- you omitted a required POINTER clause from a supplement definition; or
- you misspelled a defined-record-name or subrecord-name in a CONTAINS clause.

DATA DEFINITION PROCESSOR OUTPUT

| | | | | | |
|---|--|-----|-------|--|---|
| ○ | LINE# | SVC | ERROR | DIAGNOSTIC MESSAGE | ○ |
| ○ | 00058 | U | 014 | SYNTAX REQUIRES ITEM-NAME/DATA-NAME, TOTAL INVALID. | ○ |
| ○ | 00058 | U | 139 | -SUSPEND CHECKING INVALID SOURCE STATEMENT ON THIS LINE. | ○ |
| ○ | 00058 | U | 140 | -RESUME CHECKING SOURCE STATEMENTS ON THIS LINE. | ○ |
| ○ | LINE# | SVC | ERROR | DIAGNOSTIC MESSAGE | ○ |
| ○ | 00020 | U | 014 | SYNTAX REQUIRES DEFINED-RECORD-NAME, DIVISION INVALID. | ○ |
| ○ | 00020 | U | 139 | -SUSPEND CHECKING INVALID SOURCE STATEMENT ON THIS LINE. | ○ |
| ○ | 00023 | U | 140 | -RESUME CHECKING SOURCE STATEMENTS ON THIS LINE. | ○ |
| ○ | 00025 | U | 014 | SYNTAX REQUIRES DEFINED-RECORD-NAME, DIVISION INVALID. | ○ |
| ○ | 00025 | U | 139 | -SUSPEND CHECKING INVALID SOURCE STATEMENT ON THIS LINE. | ○ |
| ○ | 00024 | U | 140 | -RESUME CHECKING SOURCE STATEMENTS ON THIS LINE. | ○ |
| ○ | LINE# | SVC | ERROR | DIAGNOSTIC MESSAGE | ○ |
| ○ | 00035 | C | 161 | CHANGE TO NEUTRAL SUPPLEMENT IS ILLEGAL. | ○ |
| ○ | 00036 | C | 163 | ADD TO NEUTRAL SUPPLEMENT IS ILLEGAL. | ○ |
| ○ | LINE# | SVC | ERROR | DIAGNOSTIC MESSAGE | ○ |
| ○ | 00005 | U | 076 | FILE EMPFILE HAS NO DATA RECORD. | ○ |
| ○ | 00006 | U | 009 | ILLEGAL CHARACTER DETECTED IN 01. | ○ |
| ○ | 00021 | U | 148 | REFERENCE TO EMPLOYEE-REC CANNOT BE RESOLVED. | ○ |
| ○ | 00026 | U | 159 | REFERENCE TO EMP-NR INVALID. | ○ |
| ○ | 00032 | U | 014 | SYNTAX REQUIRES ITEM-DEFINITION, EXIT PROGRAM INVALID. | ○ |
| ○ | THE DATA DEFINITION RECORD COULD NOT BE CREATED. ERROR TESTING WAS NOT COMPLETED. PLEASE CORRECT AND RECOMPILE. | | | | |
| ○ | MULTDA DATA DEFINITION COMPLETE START 20:53:26 END 20:53:50 | | | | |
| ○ | UPSI SETTING = 00 UNSUCCESSFUL COMPILATION | | | | |

Figure 5-4. Data Definition Processor Listings from Unsuccessful Runs

PART 3. UNIQUE



6. Introduction to UNIQUE

6.1. SUMMARY OF UNIQUE COMMANDS

The UNIQUE commands are:

OPEN

Initiates the UNIQUE transaction and opens a dialog with a file.

CLOSE

Ends the UNIQUE transaction.

DISPLAY

Displays the contents of a record.

NEXT

Selects the next identifier from the most recent DISPLAY, DELETE, ADD, or CHANGE command and performs the same function.

DELETE

Displays a record, which you can then delete by entering the OK command.

OK

Completes an update function – DELETE, ADD, or CHANGE.

INTRODUCTION TO UNIQUE

CANCEL

Cancels an update command – DELETE, ADD, or CHANGE.

ADD

Initiates a series of inputs and responses that result in adding a record to the file.

CHANGE

Initiates a series of inputs and responses that result in changing a record.

LIST

Lists all or selected portions of a file and performs statistical functions.

MORE

Displays the next screenful of data from the previous LIST or DETAIL command.

DETAIL

Gives a secondary listing without interrupting LIST command processing.

SHOW

Displays the format of records in the defined file, the most recent LIST and DETAIL commands, and any outstanding DISPLAY, DELETE, ADD, or CHANGE command.

The UNIQUE commands are described in Section 7 with examples of their use.

6.2. PASSWORDS AND UNIQUE

Need to access defined files

To access a defined file with UNIQUE, you must know the password for that file. You use the password in the OPEN command. There are two ways to create passwords:

Defining in data definition

1. You can define a password in the data definition (3.7). In that case, the password is the same as the defined file name. When you define a password in the data definition, all configured terminals can use that password.

Defining with NAMEREC utility

2. The IMS administrator can define a password with the NAMEREC file utility. The administrator can restrict a password to specific terminals and can change passwords, even voiding a password that was defined in the data definition. (The NAMEREC file utility is described in the IMS system support functions user guide, UP-8364 (current version).)

6.3. UNIQUE DIALOG

UNIQUE transaction

A UNIQUE dialog is a series of commands and responses dealing with a particular defined file. You can have several dialogs with different files in the same UNIQUE transaction. The first OPEN command you enter starts the UNIQUE transaction and also opens a dialog with a defined file. Additional OPEN commands close the dialog with the current defined file and open dialogs with other defined files, but the UNIQUE transaction does not end until you enter a CLOSE command.



7. UNIQUE Commands

7.1. UNIQUE FORMATS AND RULES FOR ENTERING COMMANDS

Display and hard-copy formats

The format for most UNIQUE commands is the same whether you enter them from a display or hard-copy terminal. We use display terminals in most of our examples. Two commands, ADD and CHANGE, have different formats for display and hard-copy terminals, and we give examples for both formats.

Uppercase and lowercase letters

You can enter UNIQUE commands in either uppercase or lowercase letters. We use lowercase letters and reverse print (white on black) for all input in our examples so you can easily differentiate between input and output. UNIQUE output is always displayed in uppercase.

Rules for entering commands

These general rules apply to all UNIQUE commands. Additional rules are given with individual commands where they apply:

1. Move cursor to home and clear the screen or press the start-of-entry key before entering any UNIQUE command. When using an IBM 3270 terminal, enter all commands from home position.
2. Enter at least one space between words except when other punctuation is required, such as commas, semicolons, and equal signs.
3. When you enter an identifier, value, or specification that contains blanks or special characters, enclose it in apostrophes. When a name contains an apostrophe, you must enter two apostrophes. For example, enter the name Barry's Garden Mart as 'Barry''s Garden Mart'.
4. Include decimal points and commas in numeric values where required. Do not enclose numeric values in apostrophes.

Additional rules

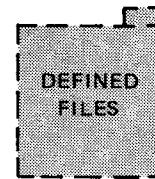
Appendix A gives more rules that apply to UNIQUE formats and commands.

EXAMPLE DATA

7.2. DATA USED IN OUR EXAMPLES

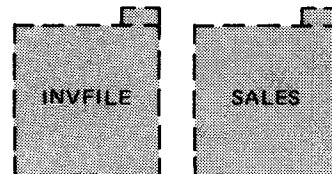
Defined file access

The files you access with UNIQUE are defined files, described in Part 2. When we use the term *file* or *files* in the UNIQUE descriptions, we always mean defined files.



Example defined files

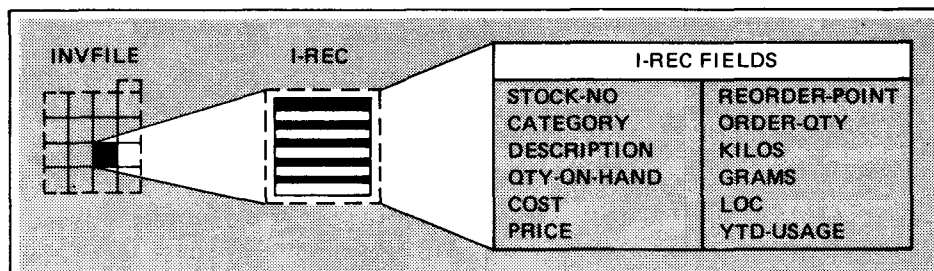
The examples in this section use two defined files – an inventory file, INVFILE, and a sales file, SALES. The data definitions for both INVFILE and SALES are in Appendix E. The passwords for both files are the same as their file names.



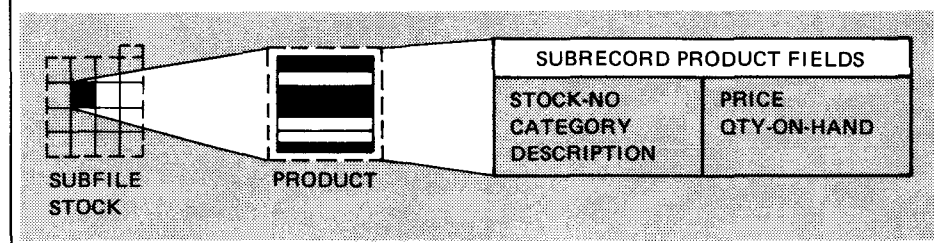
INVFILE File

INVFILE is a simple defined file containing one type of defined record, called I-REC. INVFILE also has a subrecord, PRODUCT, which is a variant of I-REC. You access PRODUCT records through a subfile called STOCK.

INVFILE file and I-REC record

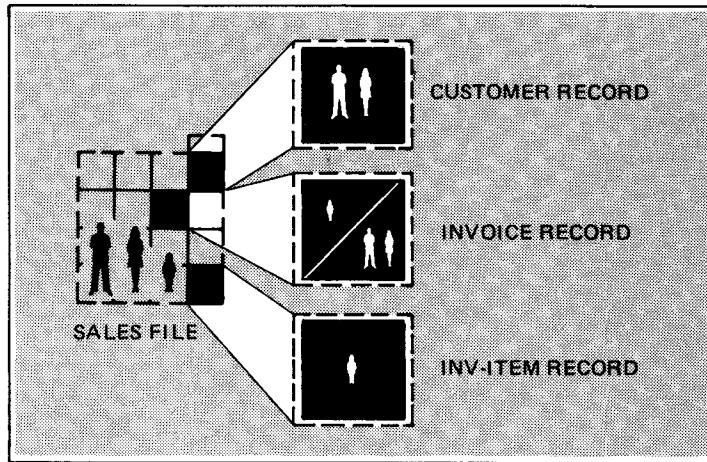


STOCK subfile and PRODUCT subrecord



SALES File

SALES is a hierarchical file containing three kinds of records. The record names are CUSTOMER, INVOICE, and INV-ITEM. For each CUSTOMER record, there are one or more INVOICE records, and for each INVOICE record, there are one or more INV-ITEM records. We call this relationship *parent-child*. CUSTOMER is a *parent* to INVOICE, INVOICE is a *child* to CUSTOMER and a *parent* to INV-ITEM, and INV-ITEM is a *child* to INVOICE.



CUSTOMER record

| CUSTOMER RECORD FIELDS | |
|------------------------|-------|
| CUST-NAME | ST |
| CUST-ID | ZIP |
| ADDRESS | PHONE |
| CITY | |

INVOICE record

| INVOICE RECORD FIELDS | |
|-----------------------|-----------|
| INV-NO | TAX-AMT |
| INV-DATE | INV-GROSS |
| TOTAL-SALES | |

INV-ITEM record

| INV-ITEM RECORD FIELDS | |
|------------------------|------------|
| NMBR | QUAN |
| ITEM-CODE | UNIT-PRICE |
| DESCRIPTION | TOT-PRICE |

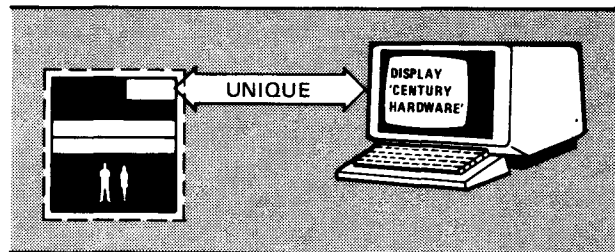
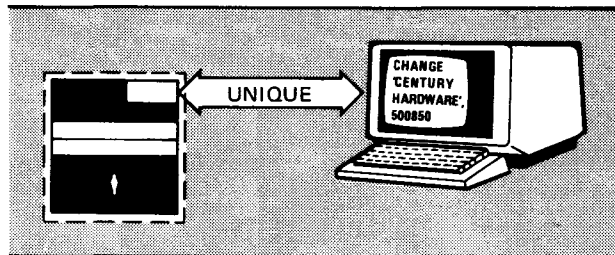
Record fields

UNIQUE uses the record fields as column headers when it displays the contents of records.

EXAMPLE DATA

Identifiers and Item Names

The first field in a record (STOCK-NO in INVFILE; CUST-NAME, INV-NO, or NMBR in SALES) is its identifier. The other record fields are called item-names. When you want to display or update a record, you name its identifier. To display or update a child record, like INVOICE or INV-ITEM, you also name the identifiers of records above it in the hierarchy. In the record display, UNIQUE links the parent record identifiers to the child record identifier. We give examples of this with the individual commands.

TO DISPLAY A PARENT RECORD**TO UPDATE A CHILD RECORD**

7.3. OPENING A UNIQUE DIALOG

Function

The OPEN command initiates a UNIQUE transaction and opens a dialog with a file. You can issue another OPEN command at any time during the transaction to access a different file. Its format is:



Format

OPEN password

Password

password

Is the password assigned to the file by the IMS administrator. It may be the actual name of the file or it may be a different name.

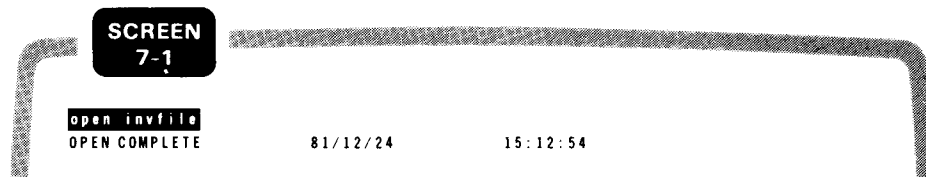
Example

The command:

```
open invfile
```

OPEN COMPLETE message

initiates a UNIQUE transaction and requests access to the inventory file, INVFILE. UNIQUE responds with an OPEN COMPLETE message, giving the date and time in hours, minutes, and seconds. After the input and response, the screen display looks like this:



Entered with another command

To save time, you can transmit the OPEN command together with one other UNIQUE command, but you do not receive the OPEN COMPLETE message. We show examples of this with the DISPLAY, LIST, and SHOW commands. (See 7.5, 7.16, and 7.25.)

7.4. ENDING THE UNIQUE TRANSACTION

Function

The CLOSE command terminates the UNIQUE transaction. Its format is:



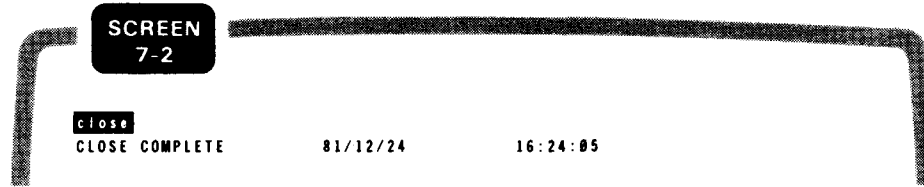
Format

CLOSE

OPEN AND CLOSE COMMANDS

CLOSE COMPLETE
message

UNIQUE responds with a CLOSE COMPLETE message, giving the date and time in hours, minutes, and seconds. After the input and response, the screen display looks like this:

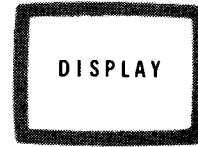
Example**Ending a dialog**

You do not need to use the CLOSE command to end a dialog with one file before starting a dialog with another file. When you issue another OPEN command, UNIQUE closes the first file and allows you to access the second file.

7.5. DISPLAYING A RECORD

Function

The DISPLAY command displays the contents of a specific record or records, with column headings. Its format is:



Format

DISPLAY identifier-1[;identifier-2]...

Displaying a Parent Record

Example

Suppose you want to display a customer record in the SALES file. You can issue the OPEN command, followed immediately by a DISPLAY command:

```
open sales
display 'century hardware'
```

After the input and response, the screen looks like this:



Displaying a Child Record

Parent-child identifier

When you want to display a child record, you give the parent record identifier first, followed by a comma and the child record identifier. (This is not the same as giving two identifier names, which we'll cover next.) Suppose you want to display invoice 500850 for Century Hardware. Enter:

Example

```
display 'century hardware',500850
```

DISPLAYING RECORDS

The input and response look like this on the screen:

```

SCREEN
  7-4
display 'century hardware'.500850
INV-NO          INV-DATE      TOTAL-SALES     TAX-AMT
CENTURY HARDWARE.500850  8/13/81        8,350.00       360.00
INV-GROSS
8,710.00
  
```

Displaying More than One Record*Using semicolons*

You can request more than one record with the same DISPLAY command. Key in a semicolon after each identifier except the last. The maximum number of identifiers you can specify on one DISPLAY command is 10.

Example

Suppose you want to display Century Hardware, Barbara's Greenery, and Barry's Garden Mart. Key in:

```
display 'century hardware'; 'barbara's greenery'; 'barry's
garden mart'
```

Embedded NEXT command

When you name more than one record, UNIQUE displays the records one at a time. UNIQUE embeds a NEXT command in the screen display for each record except the last:

Example

```

SCREEN
  7-5
display 'century hardware'; 'barbara's greenery'; 'barry's garden mart'
                                △NEXT□
CUST-NAME          CUST-ID  ADDRESS                CITY          ST  ZIP
CENTURY HARDWARE   5-60814  11580 AIRPORT BLVD    KALAMAZOO     MI  67590
PHONE
950-447-5312
  
```

When you press the TRANSMIT key, UNIQUE displays the next record.

Replacing Identifiers with Hyphens

When you request more than one child record, you don't have to repeat the names of the parent records. For all records except the first, enter hyphens in place of the parent record names. Also omit the commas. For instance, suppose you want to see items 01, 02, and 03 on invoice 500850 for Century Hardware. Enter:

Example

```
display 'century hardware',500850,01;--02;--03
```

For items 02 and 03, the first hyphen represents Century Hardware, the second represents invoice 500850.

The input and response look like this:

SCREEN 7-6

```
display 'century hardware',500850,01;--02;--03
```

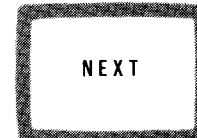
| NMBR | ITEM-CODE | DESCRIPTION | QUAN | UNIT-PRICE | TOTAL |
|----------------------------|-----------|------------------|------|------------|--------|
| CENTURY HARDWARE.500850.01 | 103010 | SAW-CIRCULAR 6IN | 6 | 25.50 | 153.00 |

△NEXT□

7.6. SELECTING THE NEXT RECORD

Function

The NEXT command selects the next identifier from the most recent DISPLAY, DELETE, ADD, or CHANGE command. The function UNIQUE performs is determined by that previous command. Its format is:



Format

NEXT

Embedded NEXT command

As we showed in the last example for the DISPLAY command, UNIQUE includes the NEXT command in the screen display when you enter more than one identifier. You simply press the TRANSMIT key (or CTRL/C on a hard-copy terminal) to display the next record.

DISPLAYING RECORDS

Displaying next record later

Sometimes you don't want to display the next record immediately. For instance, you might decide to enter another UNIQUE command such as LIST or SHOW. You can display the next record later by keying in the NEXT command. However, you must not enter another DISPLAY, DELETE, ADD, or CHANGE command in between.

Example

For example, if you want to see the next record from the last DISPLAY command and the DISPLAY screen is no longer in effect, key in:

next

Because still another record remains to be displayed, UNIQUE embeds another NEXT command in the response:

SCREEN 7-7

next

| NMBR | ITEM-CODE | DESCRIPTION | QUAN | ΔNEXT |
|----------------------------|-----------|--------------|------|-------|
| CENTURY HARDWARE.500050.02 | 122040 | TOOLBOX-23IN | 6 | |
| UNIT-PRICE | TOTAL | | | |
| 33.99 | 203.94 | | | |

When you press the TRANSMIT key, UNIQUE displays the third record you requested:

SCREEN 7-8

| NMBR | ITEM-CODE | DESCRIPTION | QUAN |
|----------------------------|-----------|--------------------|------|
| CENTURY HARDWARE.500050.03 | 573636 | SCREW DRIVER-PH9IN | 24 |
| UNIT-PRICE | TOTAL | | |
| 1.95 | 46.80 | | |

DISPLAY COMPLETE message

Notice that NEXT does not appear on the screen because 03 is the last identifier. However, you can enter the NEXT command and receive a DISPLAY COMPLETE message:

SCREEN 7-9

next

| | | |
|------------------|----------|----------|
| DISPLAY COMPLETE | 81/12/24 | 16:54:06 |
|------------------|----------|----------|

7.7. DELETING A RECORD

Function

The DELETE command lets you delete a record after viewing its contents. UNIQUE displays the record you specify on the DELETE command and then deletes it after you issue an OK command. Its format is similar to the DISPLAY command:



Format

DELETE identifier-1[;identifier-2]...

Update state

The DELETE command places your terminal in an update state. While the terminal is in this state, UNIQUE does not accept any other commands until you issue an OK or CANCEL command.

Deleting One Record

Example

Suppose you want to delete item 03 on invoice 500850 for Century Hardware. Key in:

```
delete 'century hardware',500850,03
```

The input and response look like this on the screen:

| SCREEN 7-10 | | | |
|--|-----------|--------------------|------|
| <code>delete 'century hardware',500850,03</code> | | | |
| NMBR | ITEM-CODE | DESCRIPTION | QUAN |
| CENTURY HARDWARE.500850.03 | 573636 | SCREW DRIVER-PH9IN | 24 |
| UNIT-PRICE | TOTAL | | |
| 1.95 | 46.80 | | |

OK command

Now key in the OK command to actually delete the record:

| SCREEN 7-11 | | |
|-----------------|----------|----------|
| <code>ok</code> | | |
| DELETE COMPLETE | 81/12/24 | 16:57:24 |

DELETE COMMAND**Canceling a Deletion**

If you decide not to delete the record after seeing its contents, issue the CANCEL command to cancel the deletion:

Example

```

SCREEN
7-12

cancel
DELETE CANCELLED      81/12/24      16:57:24
  
```

Deleting More than One Record*Example*

As with the DISPLAY command, you can request more than one record at a time with the same DELETE command. The maximum number of identifiers you can specify on one DELETE command is 10. To delete invoice items 02 and 03, key in:

```
delete 'century hardware'.500850.02;--03
```

UNIQUE displays the first record you request. The NEXT command is not embedded in the response to the DELETE command, but is embedded in the response to the OK or CANCEL command. The sequence of inputs and responses to delete both records is:

```

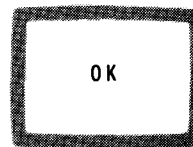
SCREEN
7-13

delete 'century hardware'.500850.02;--03
NMBR          ITEM-CODE  DESCRIPTION          QUAN
CENTURY HARDWARE.500850.02  122040  TOOLBOX-23IN          6
UNIT-PRICE    TOTAL
  33.99        203.94

ok
                                     .NEXT
DELETE COMPLETE      81/12/24      16:54:29
NMBR  ITEM-CODE  DESCRIPTION          QUAN  UNIT-PRICE    TOTAL
  03  573636    SCREW DRIVER-PH9IN    24    1.95         46.80
UNIT-PRICE    TOTAL
  1.95         46.80

ok
DELETE COMPLETE      81/12/24      16:56:05
  
```


7.8. AUTHORIZING AN UPDATE FUNCTION



Function The OK command authorizes UNIQUE to execute the updating function you requested with the previous command. You always need to use it with the DELETE command and with the hard-copy form of the ADD and CHANGE commands. You use it under certain circumstances with the display form of the ADD and CHANGE commands. Its format is:

Format OK

Completion message UNIQUE responds with a message giving you the date and time the update was completed.

Hard-copy terminal requirement Before keying in the OK command from a hard-copy terminal, you must press the carriage return key and the line feed key. If you fail to do this, UNIQUE cancels the update operation and returns an INPUT ALTERED message to the terminal. You must then reenter the DELETE, ADD, or CHANGE command.

Embedded NEXT command When you specify more than one record on the DELETE, ADD, or CHANGE command, UNIQUE embeds a NEXT command in the response to the OK command. See the DELETE, ADD, and CHANGE commands for examples of the OK command.

7.9. CANCELING AN UPDATE FUNCTION



Function The CANCEL command cancels the update function you requested with the previous DELETE, ADD, or CHANGE command. Its format is:

Format CANCEL

Cancellation message UNIQUE responds with a message giving you the date and time of the cancellation.

OK AND CANCEL COMMANDS

*Hard-copy terminal
requirement*

Before keying in the CANCEL command from a hard-copy terminal, you should press the carriage return key and the line feed key. If you fail to do this, UNIQUE returns an INPUT ALTERED message. In either case, UNIQUE cancels the update operation.

*Embedded NEXT
command*

When you specify more than one record on the DELETE, ADD, or CHANGE command, UNIQUE embeds a NEXT command in the response to the CANCEL command. See the DELETE, ADD, and CHANGE commands for examples of the CANCEL command.

7.10. ADDING A RECORD



| | |
|--|---|
| <i>Function</i> | The ADD command initiates a series of inputs and responses that result in adding a record to the file. |
| <i>Update state</i> | The ADD command places your terminal in an update state. While your terminal is in this state, UNIQUE does not accept any other commands except OK or CANCEL. |
| <i>Two formats</i> | There are two different formats for this command – the display format and the hard-copy format. The entire update sequence is different for the two formats, so we treat them as two separate commands. |
| <i>Display format use</i> <i>Hard-copy format use</i> | You can use the display format at any display terminal except an IBM 3270 display station. The hard-copy format is intended for hard-copy terminals, but you can use it at any terminal. |

7.11. DISPLAY FORMAT OF THE ADD COMMAND

| | |
|------------------------------------|---|
| <i>Description</i> | In the display format of the ADD command, you identify the record you want to add. UNIQUE displays an update format, allowing you to fill in values for the items you want to add in the record. Its format is similar to the DISPLAY and DELETE commands: |
| <i>Format</i> | ADD identifier-1[;identifier-2]... |
| <i>Parent-child identifier</i> | As with the DISPLAY and DELETE commands, an identifier may be made up of parent and child identifiers (separated by commas), and you can request up to 10 records on the same ADD command. See the DISPLAY and DELETE commands for examples of these functions. |
| <i>Requesting multiple records</i> | |
| <i>Update format display</i> | In response to the ADD command, UNIQUE displays column headers and update formats (containing asterisks) for the items in the record. You overwrite the update formats with values and transmit the screen. If UNIQUE finds no errors, it adds the record to the file and displays an ADD COMPLETE message. |

ADD COMMAND

Example

Suppose you want to add a record to the inventory file, INVFILE. First, enter the OPEN command for INVFILE, then the ADD command. You need not issue a CLOSE command for the SALES file before starting a dialog with INVFILE. The input and response look like this:

```

SCREEN
 7-14

open invfile
add 111111

STOCK-NO  CATEGORY      DESCRIPTION      QTY-ON-HAND
111111    .....          .....          .....

COST      PRICE      REORDER-POINT  ORDER-QTY  KILOS  GRAMS
.....    .....          .....          .....    .....

LOC  YTD-USAGE
....  ....

< >
    
```

Transmitting update screen

Press the tab key to move the cursor to the beginning of each update format, enter a value for each item in the new record, and transmit the screen. Be sure to overwrite leading asterisks in each item with blanks or zeros. UNIQUE does not accept items with an asterisk in the first position. Place the cursor between the special characters < > to transmit the entire screen:

```

SCREEN
 7-15

open invfile
add 111111

STOCK-NO  CATEGORY      DESCRIPTION      QTY-ON-HAND
111111    saw.....      circular 12in.....      00 0200

COST      PRICE      REORDER-POINT  ORDER-QTY  KILOS  GRAMS
00.050 00  00.086 50  00.090  00.050  052  010

LOC  YTD-USAGE
e435  00.000

< >
    
```

Update validation

Error display

UNIQUE checks the values you enter against criteria established in the data definition - type of data (alphabetic, alphanumeric, or numeric), field length, and value ranges. If you enter invalid data or a value outside the allowable range, UNIQUE displays question marks in place of the update format for the invalid item. In this case, the quantity-on-hand value is incorrect because it extends beyond the length of the update format:

**SCREEN
7-16**

```

add 111111
STOCK-NO  CATEGORY      DESCRIPTION      QTY-ON-HAND
111111    SAW             CIRCULAR 12IN  77.777
COST      PRICE          REORDER-POINT  ORDER-QTY  KILOS  GRAMS
00.050.00 00.086.50      00.090        00.050    052   010
LOC      YTD-USAGE
E435    00.000
< >

```

Correcting errors

Overwrite the error field with the correct value and transmit the screen again. If all the values are now correct, UNIQUE adds the new record to the file and responds with an ADD COMPLETE message:

**SCREEN
7-17**

```

add 111111
STOCK-NO  CATEGORY      DESCRIPTION      QTY-ON-HAND
111111    SAW             CIRCULAR 12IN  00.200
COST      PRICE          REORDER-POINT  ORDER-QTY  KILOS  GRAMS
00.050.00 00.086.50      00.090        00.050    052   010
LOC      YTD-USAGE
E435    00.000
<N>

```

**SCREEN
7-18**

```

add 111111
ADD  COMPLETE  82/01/05  10:35:00

```

ADD COMMAND

Omitting items

When you omit an item, the result depends on if that field is defined as a MUST ADD item in the data definition for this file. If the item is not defined as MUST ADD, UNIQUE simply adds the record to the file without a value for the omitted item:

Nonrequired items

SCREEN 7-19

```

add 111111
STOCK-NO  CATEGORY          DESCRIPTION          QTY-ON-HAND
111111    saw              circular 12in      00.200
COST      PRICE            REORDER-POINT  ORDER-QTY  KILOS  GRAMS
00.050.00 00.086.50      00.090        00.050     052    010
LOC      YTD-USAGE
e435     .....
```

<□>

SCREEN 7-20

```

add 111111
ADD  COMPLETE  82/01/05  10:35:35
```

MUST ADD items

However, if the item was defined as MUST ADD, UNIQUE displays question marks in place of the update format for the omitted item. You must supply the missing item before UNIQUE can add the new record to the file:

SCREEN 7-21

```

add 111111
STOCK-NO  CATEGORY          DESCRIPTION          QTY-ON-HAND
111111    saw              circular 12in      00.200
COST      PRICE            REORDER-POINT  ORDER-QTY  KILOS  GRAMS
.....    00.086.50      00.090        00.050     052    010
LOC      YTD-USAGE
e435     .....
```

<□>

**SCREEN
7-22**

```
add 111111
STOCK-NO CATEGORY      DESCRIPTION      QTY-ON-HAND
111111  SAW             CIRCULAR 12IN          00.200
COST      PRICE      REORDER-POINT ORDER-QTY  KILOS  GRAMS
77.777.77 00.086.50      00.090   00.050   052    010
LOC      YTD-USAGE
E435     *.*.*.*
< >
```

OK command

During the ADD sequence, you can enter the OK command to add an incomplete record, as long as you have entered all MUST ADD items. For instance, after receiving an error on the quantity on hand, suppose you decide to add the record without an entry for that field (assuming that the quantity on hand is not a MUST ADD item). This is the sequence of inputs and responses:

**SCREEN
7-23**

```
add 111111
STOCK-NO CATEGORY      DESCRIPTION      QTY-ON-HAND
111111  SAW             CIRCULAR 12IN          77.777
COST      PRICE      REORDER-POINT ORDER-QTY  KILOS  GRAMS
00.050.00 00.086.50      00.090   00.050   052    010
LOC      YTD-USAGE
E435     00.000
< >
OK
```

**SCREEN
7-24**

```
add 111111
ADD  COMPLETE  82/01/05  10:36:10
```

ADD COMMAND

If you enter the OK command and a MUST ADD item is missing, the ADD command is canceled:

```

SCREEN
  7-25

add 111111
STOCK-NO  CATEGORY      DESCRIPTION      QTY-ON-HAND
111111    SAW            CIRCULAR 12IN      00.200
COST      PRICE      REORDER-POINT  ORDER-QTY  KILOS  GRAMS
??,???.?? 00,086.50      00.125        00.050     052    010
LOC      YTD-USAGE
E435     00.000
< >
OK

```

```

SCREEN
  7-26

add 111111
ADD  CANCELLED  82/01/05  10:38:00

```

*Add nonzero data
for MUST ADD items*

Note that you must add nonzero data for MUST ADD items. UNIQUE does not display question marks on the update format when you enter zeros for a MUST ADD item. However, when you enter the OK command, UNIQUE cancels the ADD command.

Canceling update

You can also enter the CANCEL command at any time during the ADD sequence to cancel the addition of the record:

```

SCREEN
  7-27

cancel
ADD  CANCELLED          82/01/05          10/35/16

```

7.12. HARD-COPY FORMAT OF THE ADD COMMAND

Description

In the hard-copy format of the ADD command, you identify the record you want to add and also enter values for the items you want to add in the record. The format is:

Format

```
ADD identifier [item-name=]value[:[item-name=]value]...
```

Parent-child identifier

As in the display format, an identifier may be made up of parent and child identifiers (separated by commas). See the DISPLAY command for examples. Unlike the display format, you cannot request more than one record on the same hard-copy format of the ADD command.

Item-names Item-names are the names of the fields in the record. You can get these names and their sequence in the record by issuing a SHOW command. The SHOW command also tells you which items you must include when you add a record. To see the record format for the examples in this section, see screen 7-65 in 7.25.

Input message length Note that the length of the input message, including identifiers, item-names, and values, is limited to 256 characters.

Output display In response to the ADD command, UNIQUE displays column headers, the values you entered, if acceptable, and error notations. You can enter additional values, and UNIQUE repeats the display. When you are satisfied with the new record, you key in the OK command to add the record to the file.

Two subformats We can illustrate the hard-copy format of the ADD command more easily by breaking it down into two formats. The first format is easier when you are entering values for only a few selected items in the record. The second format is easier when you are entering a value for every item in the record. You can also use a combination of the two formats.

Naming items and values

1. In the first format, you name each item and its value:

Format

```
ADD identifier item-name=value[;item-name=value]...
```

When you use this format, you can enter the items in any order. Enter a semicolon after each value except the last. No spaces are allowed before or after the equal sign or before the semicolon. A space after the semicolon is optional. Here's an example:

Example

```
add 111111 category=saw;cost=50.00; price=86.50
```

Entering values by position

2. In the second format, you omit the item-names and enter values for the items in the exact order they appear in the record (the same order as the column headers):

Format

```
ADD identifier value[;value]...
```

Here's an example:

Example

```
add 111111 saw;'circular 12in';200;50.00;86.50
```

ADD COMMAND

Omitting items

When you don't want to enter a value for every item in the record, you can omit an item by entering a semicolon in its position. For instance, to enter values only for category, cost, and price, key in this format:

Example

add 111111 saw;;;50.00;86.50

The first semicolon after SAW is the separator. The second and third semicolons replace values for DESCRIPTION and QTY-ON-HAND.

Item-name/positional combination

3. You can use a combination of the two formats, like this:

Formats

ADD identifier item=value[:value]...

or

ADD identifier value[:item=value]...

UNIQUE determines value positions from the last item-name you enter. For example, to enter values for CATEGORY, COST, and PRICE:

Examples

add 111111 category=saw;cost=50.00;86.50

or

add 111111 saw;cost=50.00;86.50

Update validation

In response to the ADD command, UNIQUE displays column headers, valid entries, and question marks for invalid entries and omitted MUST ADD items. As with the display format, UNIQUE checks for data type and value ranges. However, UNIQUE does not check for field length. If you enter a value with too many characters, UNIQUE truncates the value. Screen 7-28 illustrates the UNIQUE response, with question marks for missing MUST ADD items.

SCREEN 7-28

```

add 111111 category=saw.cost=50.00;86.50
STOCK-NO  CATEGORY      DESCRIPTION          QTY-ON-HAND
111111    SAW              ??????????????????  **.*
COST      PRICE      REORDER-POINT  ORDER-QTY  KILOS  GRAMS
00.050.00 00.086.50      77.777          **.*      **.*
LOC      YTD-USAGE
?????    **.*
    
```

Correcting errors

You make corrections by entering either an item-name and value or just a value. To enter a value without the item-name, you must key in the correct number of semicolons to place the value in its correct position in the record. For example, to enter values for DESCRIPTION and QTY-ON-HAND, key in either format:

Item-name example

description='circular 12in';qty-on-hand=200

or

Positional example

::'circular 12in';200

The second method can become rather cumbersome when you want to enter a value that isn't near the beginning of the record. For instance, to enter a value for LOC, you would have to count the number of item-names preceding LOC and key in this format:

::::::::::::e435

OK command

Canceling update

When you are satisfied with the new record, key in the OK command to add the record to the file. If you decide not to add the record, key in the CANCEL command. If you key in the OK command and a MUST ADD item is missing, UNIQUE cancels the ADD command. Here's an example of the entire sequence of inputs and outputs:

SCREEN 7-29

```
add 111111 category=saw;cost=50.00;86.50
```

| STOCK-NO | CATEGORY | DESCRIPTION | QTY-ON-HAND |
|-----------|-----------|----------------------|-------------|
| 111111 | SAW | ???????????????????? | |
| COST | PRICE | REORDER-POINT | ORDER-QTY |
| 00.050.00 | 00.086.50 | ??,??? | ..,... |
| LOC | YTD-USAGE | | |
| ???? | ..,... | | |

```
description='circular 12in';reorder-point=90;loc=e435
```

| STOCK-NO | CATEGORY | DESCRIPTION | QTY-ON-HAND |
|-----------|-----------|---------------|-------------|
| 111111 | SAW | CIRCULAR 12IN | |
| COST | PRICE | REORDER-POINT | ORDER-QTY |
| 00.050.00 | 00.086.50 | 00.090 | ..,... |
| LOC | YTD-USAGE | | |
| E435 | ..,... | | |

```
ok
```

ADD COMPLETE 82/01/05 13:12:26

CHANGE COMMAND**7.13. CHANGING A RECORD***Function*

The CHANGE command initiates a series of inputs and responses that result in changing one or more

*Update state*

The CHANGE command places your terminal in an update state. While your terminal is in this state, UNIQUE does not accept any other commands except OK or CANCEL.

*Display format use**Hard-copy format use*

There are two different formats for this command: the display format and the hard-copy format. The entire update sequence is different for the two formats, so we treat them as two separate commands.

You can use the display format at any display terminal except an IBM 3270 display station. The hard-copy format is intended for hard-copy terminals, but you can use it at any terminal.

7.14. DISPLAY FORMAT OF THE CHANGE COMMAND*Description*

In the display format of the CHANGE command, you identify the record you want to change. UNIQUE displays an update format, allowing you to fill in values for the items you want to change in the record. Its format is similar to the DISPLAY, DELETE, and ADD commands:

Format

CHANGE identifier-1[;identifier-2]...

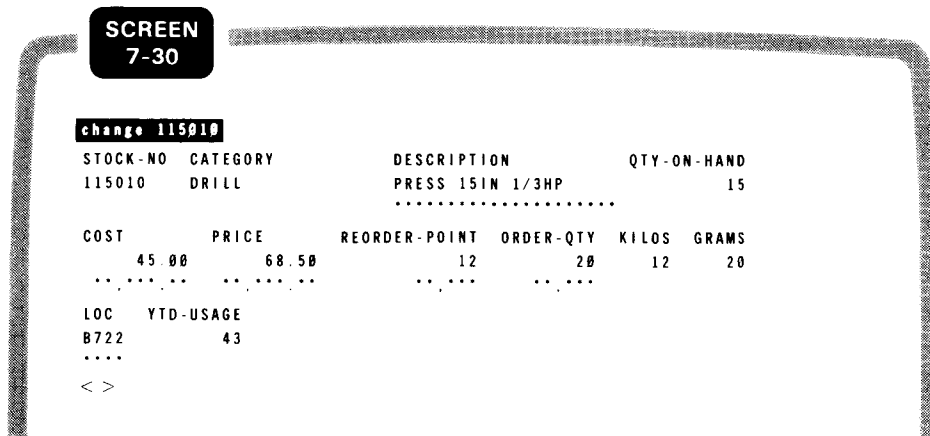
Parent-child identifier

As with the DISPLAY, DELETE, and display format ADD commands, an identifier may be made up of parent and child identifiers (separated by commas), and you can request up to 10 records on the same CHANGE command. See the DISPLAY and DELETE commands for examples of these functions.

*Requesting multiple records**Update format display*

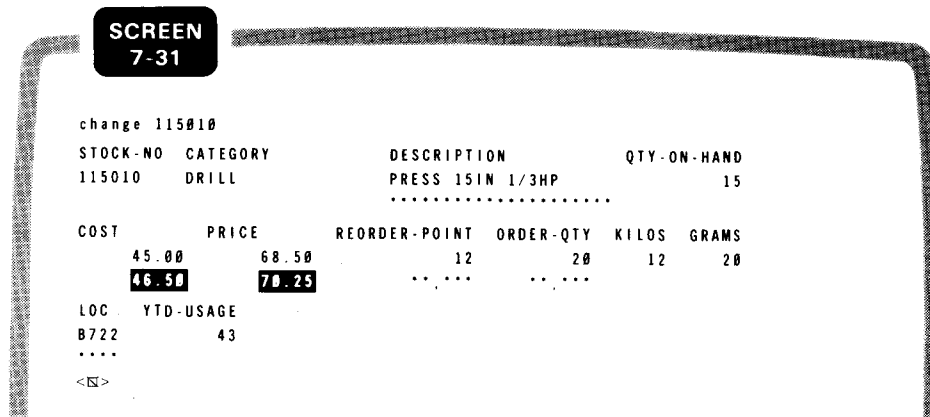
In response to the CHANGE command, UNIQUE displays column headers and the current value of each item in the record. Below the current values, UNIQUE displays update formats (containing asterisks) for those items that you are allowed to change. Items that do not have update formats cannot be changed. Suppose you want to change record 115010 in the inventory file:

Example



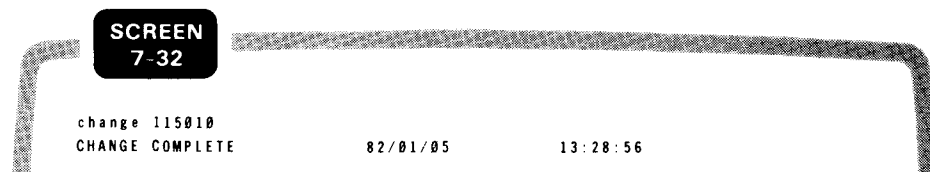
Transmitting update screen

Press the tab key to move the cursor to the beginning of each update format and enter new values only for those items you want to change. Be sure to overwrite leading asterisks in each item with blanks or zeros. UNIQUE does not accept new values with an asterisk in the first position. Transmit the screen by placing the cursor between the special characters < >:



Update validation

UNIQUE checks the values you enter against criteria established in the data definition – type of data (alphabetic, alphanumeric, or numeric), field length, and value ranges. If there are no errors, UNIQUE replaces the old record in the file with the changed record and displays a CHANGE COMPLETE message:



When another terminal is updating record

Note, however, that if the record is in the process of being updated by a transaction at another terminal, UNIQUE does not make the requested change and an INVALID REQUEST message is displayed. Reenter the CHANGE command.

CHANGE COMMAND

Error display

If you enter invalid data or a value outside the allowable range, UNIQUE displays question marks in place of the update format for the invalid item. For example, suppose you key in an o instead of a zero in the reorder-point item:

SCREEN 7-33

change 115010

| STOCK-NO | CATEGORY | DESCRIPTION | QTY-ON-HAND | | | |
|----------|-----------|------------------|-------------|-------|-------|----|
| 115010 | DRILL | PRESS 15IN 1/3HP | 15 | | | |
| | | | | | | |
| COST | PRICE | REORDER-POINT | ORDER-QTY | KILOS | GRAMS | |
| 45.00 | 68.50 | 12 | 20 | 12 | 20 | |
| 46.50 | 70.25 | ..o | .. | .. | .. | .. |
| LOC | YTD-USAGE | | | | | |
| B722 | 43 | | | | | |
| | | | | | | |

< >

SCREEN 7-34

change 115010

| STOCK-NO | CATEGORY | DESCRIPTION | QTY-ON-HAND | | | |
|----------|-----------|------------------|-------------|-------|-------|----|
| 115010 | DRILL | PRESS 15IN 1/3HP | 15 | | | |
| | | | | | | |
| COST | PRICE | REORDER-POINT | ORDER-QTY | KILOS | GRAMS | |
| 45.00 | 68.50 | 12 | 20 | 12 | 20 | |
| 46.50 | 70.25 | ???.??? | .. | .. | .. | .. |
| LOC | YTD-USAGE | | | | | |
| B722 | 43 | | | | | |
| | | | | | | |

< >

OK command

You can, if you want, enter the OK command to change the valid items in the record. Any item for which you entered an incorrect value remains unchanged:

SCREEN 7-35

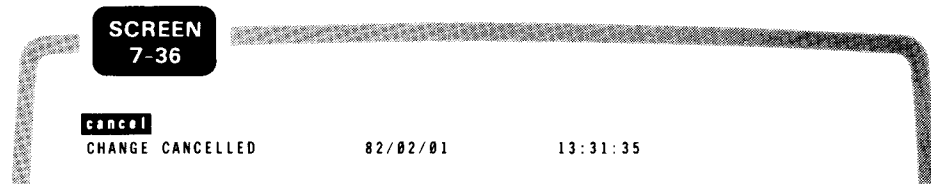
OK

CHANGE COMPLETE 82/01/05 13:31:16

Again, if the record is being updated at another terminal, UNIQUE does not change the record but displays an INVALID REQUEST message.

Canceling update

You can also enter the CANCEL command at any time during the CHANGE sequence to cancel the update:



7.15. HARD-COPY FORMAT OF THE CHANGE COMMAND

Description

In the hard-copy format of the CHANGE command, you identify the record you want to change and also enter values for the items you want to change in the record. Its format is:

Format

```
CHANGE identifier [item-name=value[;item-name=value]]...
```

Parent-child identifier

As in the display format, an identifier may be made up of parent and child identifiers (separated by commas). See the DISPLAY command for examples. Unlike the display format, you cannot request more than one record on the same hard-copy format CHANGE command.

Item-names

Item-names are the names of the fields in the record. You can get these names and their sequence in the record by issuing a SHOW command. The SHOW command also tells you which items you are allowed to change in the record. To see the record format for the examples in this section, see screen 7-65 in 7.25.

Output display

In response to the CHANGE command, UNIQUE displays column headers, the values you entered, if acceptable, and error notations. You can enter additional values, and UNIQUE repeats the display. When you are satisfied with the new record, you key in the OK command to change the record in the file.

Two subformats

We can break down the hard-copy format of the CHANGE command into two formats, as we did with the ADD command:

Naming items and values

1. In the first format, you name each item and its value:

Format

```
CHANGE identifier item-name=value[;item-name=value]...
```

When you use this format, you can enter the items in any order. Enter a semicolon after each value except the last. No spaces are allowed before or after the equal sign or before the semicolon. A space after the semicolon is optional. Here's an example:

Example

```
change 115010 cost=46.50;price=70.25
```

CHANGE COMMAND*Entering values by position*

2. In the second format, you omit the item-names and enter values for the items in the exact order they appear in the record (the same order as the column headers):

Format

```
CHANGE identifier value[;value]...
```

When you omit an item, enter a semicolon in its position. For example:

Example

```
change 115010 ;;;46.50;70.25
```

In this example, the three semicolons replace values for CATEGORY, DESCRIPTION, and QTY-ON-HAND.

Item-name/positional combination

3. You can use a combination of the two formats, like this:

Formats

```
CHANGE identifier item=value[;value]...
```

or

```
CHANGE identifier value[;item=value]...
```

UNIQUE determines value positions from the last item-name you enter. For example:

Example

```
change 115010 cost=46.50;70.25
```

Update validation

In response to the CHANGE command, UNIQUE displays column headers, the values you specify, and question marks for invalid entries. As with the display format, UNIQUE checks for data type and value ranges. However, UNIQUE does not check for field length. If you enter a value with too many characters, UNIQUE truncates the value.

*Field length not checked**When change is not allowed*

Remember that you can enter values only for items you are allowed to change in the record. If you enter a value for an item that cannot be changed, UNIQUE does not recognize the item-name. Instead, UNIQUE interprets the entry as a positional value string and attempts to change the next field for which change is allowed.

Example

For example, in screen 7-37, an entry is made for QTY-ON-HAND. Change is not allowed for QTY-ON-HAND (see screen 7-65). UNIQUE interprets the entire entry, QTY-ON-HAND=58 as a value for the next item after PRICE REORDER-POINT. But REORDER-POINT is defined in the data definition as a numeric item, so the value is rejected and UNIQUE displays question marks.

SCREEN
7-37

```
change 115010 cost=46.50 price=70.25 qty-on-hand=58
```

| STOCK-NO | CATEGORY | DESCRIPTION | QTY-ON-HAND | | |
|-----------|-----------|------------------|-------------|-------|-------|
| 115010 | DRILL | PRESS 15IN 1/3HP | 15 | | |
| COST | PRICE | REORDER-POINT | ORDER-QTY | KILOS | GRAMS |
| 00,046.50 | 00,070.25 | ??,??? | 20 | 12 | 20 |
| LOC | YTD-USAGE | | | | |
| B722 | 43 | | | | |

Correcting errors

You make corrections by entering either an item-name and value or just a value. To enter a value without the item-name, you must key in the correct number of semicolons to place the value in its correct position in the record. For example, to enter the correct value for REORDER-POINT, use either of these formats to key in data:

reorder-point=12

or

;;;;;12

OK command

Canceling update

When you are satisfied with the changed record, key in the OK command to change the record in the file. If you decide not to change the record, key in the CANCEL command. Here's an example of the entire sequence of inputs and outputs:

SCREEN
7-38

```
change 115010 cost=46.50 price=70.25 qty-on-hand=58
```

| STOCK-NO | CATEGORY | DESCRIPTION | QTY-ON-HAND | | |
|-----------|-----------|------------------|-------------|-------|-------|
| 115010 | DRILL | PRESS 15IN 1/3HP | 15 | | |
| COST | PRICE | REORDER-POINT | ORDER-QTY | KILOS | GRAMS |
| 00,046.50 | 00,070.25 | ??,??? | 20 | 12 | 20 |
| LOC | YTD-USAGE | | | | |
| B722 | 43 | | | | |

```
reorder-point 12
```

| STOCK-NO | CATEGORY | DESCRIPTION | QTY-ON-HAND | | |
|-----------|-----------|------------------|-------------|-------|-------|
| 115010 | DRILL | PRESS 15IN 1/3HP | 15 | | |
| COST | PRICE | REORDER-POINT | ORDER-QTY | KILOS | GRAMS |
| 00,046.50 | 00,070.25 | 12 | 20 | 12 | 20 |
| LOC | YTD-USAGE | | | | |
| B722 | 43 | | | | |

```
ok
```

CHANGE COMPLETE 82/01/05 14:05:50

When another terminal is updating record

Note, however, that if the record is in the process of being updated by a transaction at another terminal, UNIQUE does not make the requested change and an INVALID REQUEST message is displayed. Reenter the CHANGE command.

LIST COMMAND

7.16. LISTING THE RECORDS IN A FILE*Function*

The LIST command displays:

- the contents of all the records in a file;
- the contents of selected records in a file; or
- specific items in all or selected records.

In addition to displaying record contents, the LIST command can calculate and display the results of statistical functions.

The format of the LIST command is:

Format

```
LIST [ [display-content-specification] ] ...
      [ [IF conditional-expression]; ]
      [FOR identifier-1]
      [ {AFTER} identifier-2 ]
      [ {FROM } ]
      [statistical-function [item-name-1[, item-name-2]...]]
```

LIST parameters

All of the parameters on the LIST command are optional. You can enter the LIST command with no additional specifications to get a complete listing of the records in a file. When you enter more than one parameter with the LIST command, you must enter them in the order shown in the format.

Because of the complexity of the LIST command format, we'll break it down into separate formats for different functions.

**7.17. LISTING COMPLETE CONTENTS OF ALL RECORDS
(UNQUALIFIED LIST)***Function*

To list the complete contents of all the records in a file, enter the LIST command with no additional specifications. This is called an unqualified LIST command:

Format

```
LIST
```

Example

Here's an example of an unqualified LIST command for the INVFILE file and the resulting display. An asterisk precedes the headers for each record type (in this case, there is only one record type), and a period precedes each record. Because the file takes up more than one screen, UNIQUE embeds a MORE LIST command in the screen display:

Embedded MORE LIST

SCREEN
7-39

list

| STOCK-NO | CATEGORY | DESCRIPTION | QTY-ON-HAND |
|-----------|-----------|------------------|-------------|
| COST | PRICE | REORDER-POINT | LOC |
| YTD-USAGE | ORDER-QTY | KILOS | GRAMS |
| * 101110 | SAW | TABLE 9IN 1 6HP | 24 |
| 139.00 | 189.88 | 25 35 96 | 100 E206 |
| 103 | | | |
| * 101115 | SAW | TABLE 10IN 2.5HP | 18 |
| 223.00 | 299.95 | 20 35 206 | 50 E210 |
| 74 | | | |
| * 101120 | SAW | TABLE 12IN 3.5HP | 21 |
| 350.00 | 499.95 | 20 30 228 | 140 E214 |
| 89 | | | |
| * 102155 | SAW | CHAIN 18IN 3.7CU | 10 |
| 172.85 | 237.99 | 10 15 20 | 30 E400 |
| 38 | | | |
| * 102160 | SAW | CHAIN 16IN 2.5CU | 13 |
| 151.85 | 207.99 | 10 15 15 | 10 E398 |
| 44 | | | |
| * 102165 | SAW | CHAIN 14IN 2.0CU | 15 |
| 120.85 | 167.99 | 15 20 14 | 70 E396 |
| 56 | | | |

END LIST

When you press the TRANSMIT key, UNIQUE displays the next screenful. On the last LIST screen, END LIST replaces MORE LIST in the upper right corner.

**7.18. SELECTING ITEMS OR RECORDS FOR LISTING
(DISPLAY-CONTENT-SPECIFICATION)**

Function

The first parameter on the LIST command selects specific record items or complete records or subrecords for listing:

Format

LIST display-content-specification [;]...

LIST COMMAND*Specification types*

There are four types of display content specifications:

1. item-names
2. record-name
3. ALL
4. subrecord-name

Selecting Record Items for Listing (Item-names)*Entering item-names*

To display specific items in each record, enter the LIST command with item-names. Separate item-names with one or more spaces. UNIQUE displays the record identifier and the items you select, with column headers. For example, suppose you want to display the category, description, and price items for each record in INVFILE:

SCREEN
7-40

| * STOCK-NO | CATEGORY | DESCRIPTION | ΔMORE | LIST□ |
|------------|----------|------------------|-------|--------|
| . 10110 | SAW | TABLE 9IN 1.6HP | | 189.88 |
| . 10115 | SAW | TABLE 10IN 2.5HP | | 299.95 |
| . 10120 | SAW | TABLE 12IN 3.5HP | | 499.95 |
| . 102155 | SAW | CHAIN 18IN 3.7CU | | 237.99 |
| . 102160 | SAW | CHAIN 16IN 2.5CU | | 207.99 |
| . 102165 | SAW | CHAIN 14IN 2.0CU | | 167.99 |
| . 102170 | SAW | CHAIN 10IN 2.0CU | | 77.99 |
| . 103010 | SAW | CIRCULAR 6IN | | 25.50 |
| . 103013 | SAW | CIRCULAR 7IN | | 34.50 |
| . 103016 | SAW | CIRCULAR 8IN | | 48.50 |
| . 115000 | DRILL | PRESS 72IN 1/2HP | | 340.94 |
| . 115010 | DRILL | PRESS 15IN 1/3HP | | 68.50 |
| . 116550 | DRILL | HAND 1/2IN 7/8HP | | 68.50 |
| . 116555 | DRILL | HAND 1/2IN 3/8HP | | 48.50 |
| . 116560 | DRILL | HAND 3/8IN 1/5HP | | 28.50 |
| . 116565 | DRILL | HAND 1/4IN 1/6HP | | 7.49 |
| . 121010 | TOOLBOX | CHEST 10 DRAWER | | 144.99 |
| . 121020 | TOOLBOX | CHEST 5 DRAWER | | 99.99 |
| . 121030 | TOOLBOX | CHEST 3 DRAWER | | 94.99 |
| . 122040 | TOOLBOX | FLAT TOP 23IN | | 33.99 |
| . 122050 | TOOLBOX | TWO COVER 17IN | | 32.75 |

Order of entry

You can enter the item-names in any order. UNIQUE displays the items in the order you select. For example, you might want to display the items in the preceding example in reverse order:

SCREEN
7-41

| list | price | description | category | △MORE | LIST□ |
|------------|--------|------------------|----------|-------|-------|
| * STOCK-NO | PRICE | DESCRIPTION | CATEGORY | | |
| . 101110 | 189.88 | TABLE 9IN 1.6HP | SAW | | |
| . 101115 | 299.95 | TABLE 10IN 2.5HP | SAW | | |
| . 101120 | 499.95 | TABLE 12IN 3.5HP | SAW | | |
| . 102155 | 237.99 | CHAIN 18IN 3.7CU | SAW | | |
| . 102160 | 207.99 | CHAIN 16IN 2.5CU | SAW | | |
| . 102165 | 167.99 | CHAIN 14IN 2.0CU | SAW | | |
| . 102170 | 77.99 | CHAIN 10IN 2.0CU | SAW | | |
| . 103010 | 25.50 | CIRCULAR 6IN | SAW | | |
| . 103013 | 34.50 | CIRCULAR 7IN | SAW | | |
| . 103016 | 48.50 | CIRCULAR 8IN | SAW | | |
| . 115000 | 340.94 | PRESS 72IN 1/2HP | DRILL | | |
| . 115010 | 68.50 | PRESS 15IN 1/3HP | DRILL | | |
| . 116550 | 68.50 | HAND 1/2IN 7/8HP | DRILL | | |
| . 116555 | 48.50 | HAND 1/2IN 3/8HP | DRILL | | |
| . 116560 | 28.50 | HAND 3/8IN 1/5HP | DRILL | | |
| . 116565 | 7.49 | HAND 1/4IN 1/6HP | DRILL | | |
| . 121010 | 144.99 | CHEST 10 DRAWER | TOOLBOX | | |
| . 121020 | 99.99 | CHEST 5 DRAWER | TOOLBOX | | |
| . 121030 | 94.99 | CHEST 3 DRAWER | TOOLBOX | | |
| . 122040 | 33.99 | FLAT TOP 23IN | TOOLBOX | | |
| . 122050 | 32.75 | TWO COVER 17IN | TOOLBOX | | |

Multiple record types

INVFILE contains only one record type, but you can use the item-names specification in the same way for a file containing several record types. UNIQUE searches all the record types for the items you specify.

We'll give additional examples of the item-names specification in the discussions of the other LIST parameters.

Selecting Complete Records for Listing (Record-name, ALL)**One record type**

With a defined file containing only one type of record, you omit the display content specification when you want to display the entire record, as in the unqualified LIST command. You can specify the record name, but this is not necessary.

Multiple record types

However, when there is more than one kind of record in the file, you may want to display only selected record types. We'll use the SALES file to illustrate the record-name and ALL specifications. The SALES file contains CUSTOMER, INVOICE, and INV-ITEM records.

LIST COMMAND*Listing one record type*

To list just one type of record, enter the LIST command with the record name. The following example opens the SALES file and displays all CUSTOMER records. You can enter the OPEN and LIST commands together:

SCREEN
7-42

open sales list customer

| * CUST-NAME PHONE | CUST-ID | ADDRESS | CITY | ΔMORE LIST ST ZIP |
|----------------------------------|---------|------------------|------------|----------------------|
| . ABLE COMPANY 675-887-9595 | 5-40590 | 2046 CLOVERLY LA | MIDDLETOWN | PA 19789 |
| . ADDLEBY, INC. 543-129-1187 | 5-21543 | 1519 MEARN'S RD | HOMETOWN | AR 54331 |
| . ADHOC, INC. 236-657-5446 | 5-13448 | 250 HOPEWELL | GREENVILLE | GA 34687 |
| . AMARK HARDWARE 625-537-9954 | 5-60814 | 29 CATANGA ST | CATANGA | FL 42970 |
| . AMWAY VARIETY 457-332-5648 | 5-15574 | 5348 MARIAN RD | SEKONDA | WI 68867 |
| . ARROWHEAD LTD. 145-054-6700 | 5-55429 | 8410 BROAD ST | BARRINGTON | MA 30558 |
| . ASTOR HARDWARE 323-776-4289 | 5-18976 | 129 MAIN ST | ASTOR | SC 25995 |
| . ATRIUM COMPANY 278-246-6785 | 5-05946 | 550 FORREST AVE | DRINKWATER | VA 32584 |
| . ATWELL VARIETY 817-236-5579 | 5-32963 | 1846 DARIEN DR | PHILA. | PA 19950 |
| . B & M HARDWARE 345-886-7896 | 5-40896 | 1226 SHADY LA | ROCKWELL | IN 45680 |

Press the TRANSMIT key to see the next screenful of CUSTOMER records.

Listing several record types

To list more than one type of record, enter the LIST command with the record names separated by semicolons. Spaces after the semicolon are optional. The following example displays the CUSTOMER and INVOICE records. The hyphen preceding each INVOICE record identifier indicates that it is subordinate to a CUSTOMER record:

SCREEN
7-43

list customer; invoice

| * CUST-NAME | | CUST-ID | ADDRESS | CITY | △MORE LIST□ | |
|------------------|--------------|-------------|------------------|------------|-------------|-------|
| PHONE | | | | | ST | ZIP |
| * -INV-NO | INV-DATE | TOTAL-SALES | TAX-AMT | INV-GROSS | | |
| . ABLE COMPANY | | 5-40590 | 2046 CLOVERLY LA | MIDDLETOWN | PA | 19789 |
| | 675-887-9595 | | | | | |
| . -357005 | 08/15/81 | 2,599.84 | 174.80 | 2,774.64 | | |
| . -444834 | 09/10/81 | 580.17 | 28.30 | 608.47 | | |
| . -455610 | 10/01/81 | 1,367.70 | 65.00 | 1,432.70 | | |
| . ADDLEBY, INC. | | 5-21543 | 1519 MEARNS RD. | HOMETOWN | AR | 54331 |
| | 543-129-1187 | | | | | |
| . -352246 | 08/04/81 | 180.00 | 16.20 | 196.20 | | |
| . -405891 | 08/30/81 | 429.00 | 35.25 | 464.25 | | |
| . -525603 | 10/05/81 | 356.00 | 29.74 | 385.74 | | |
| . ADHOC, INC. | | 5-13448 | 250 HOPEWELL | GREENVILLE | GA | 34687 |
| | 236-657-5446 | | | | | |
| . -350208 | 08/01/81 | 3,560.00 | 143.30 | 3,603.30 | | |
| . AMARK HARDWARE | | 5-60814 | 29 CATANGA ST | CATANGA | FL | 42970 |
| | 625-537-9954 | | | | | |
| . -485465 | 09/17/81 | 846.00 | 39.45 | 885.45 | | |
| . -556347 | 10/24/81 | 2,175.00 | 136.00 | 2,311.00 | | |

Listing all record
types

To list all three record types, you can enter the command:

list customer; invoice; item

You'd get the same listing by entering an unqualified LIST:

SCREEN
7-44

list

| * CUST-NAME | | CUST-ID | ADDRESS | CITY | △MORE LIST□ | |
|----------------|--------------|--------------------|------------------|------------|-------------|-------|
| PHONE | | | | | ST | ZIP |
| * -INV-NO | INV-DATE | TOTAL-SALES | TAX-AMT | INV-GROSS | | |
| * --NMBR | ITEM-CODE | DESCRIPTION | QUAN | UNIT-PRICE | TOT-PRICE | |
| . ABLE COMPANY | | 5-40590 | 2046 CLOVERLY LA | MIDDLETOWN | PA | 19789 |
| | 675-887-9595 | | | | | |
| . -357005 | 08/15/81 | 2,599.84 | 174.80 | 2,774.64 | | |
| . --01 | 121010 | TOOLBOX-CHEST 10DR | 6 | 144.99 | 869.94 | |
| . --02 | 103010 | SAW-CIRCULAR 6IN | 12 | 25.50 | 306.00 | |
| . --03 | 102165 | SAW-CHAIN 14IN | 4 | 167.99 | 671.96 | |
| . --04 | 115000 | DRILL-PRESS 72IN | 1 | 340.94 | 340.94 | |
| . --05 | 116550 | DRILL-HAND 7/8HP | 6 | 68.50 | 411.00 | |
| . -444834 | 09/10/81 | 580.17 | 28.30 | 608.47 | | |
| . --01 | 301110 | LAWN-MOWER ELEC | 3 | 193.39 | 580.17 | |
| . -455610 | 10/01/81 | 1,367.70 | 65.00 | 1,432.70 | | |
| . --01 | 101110 | SAW-TABLE 9IN | 2 | 189.88 | 379.76 | |
| . --02 | 103016 | SAW-CIRCULAR 8IN | 8 | 48.50 | 388.00 | |
| . --03 | 121020 | TOOLBOX-CHEST 5DR | 6 | 99.99 | 599.94 | |

LIST COMMAND

The two hyphens preceding the INV-ITEM records indicate that there are two records above them in the hierarchy - CUSTOMER and INVOICE.

ALL specification

The ALL specification has the same effect as record-name - it displays all the items in a particular record. Because you do not name the record, UNIQUE determines which record type you want by the position of ALL in the LIST command. For example, to display only the CUSTOMER records, you can enter:

```
list all (equivalent to list customer)
```

To display the CUSTOMER and INVOICE records, enter:

```
list all;all (equivalent to list customer;invoice)
```

Omitting record types

Because the ALL specification is positional, you must enter a semicolon in place of any record type you skip. For example, to display only the INVOICE records, enter:

```
list ;all (equivalent to list invoice)
```

To display only the INV-ITEM records, enter:

```
list ;;all (equivalent to list inv-item)
```

Listing child records

Because INVOICE and INV-ITEM are child records, you normally would not list them without also listing the parent records. You can display the INVOICE records for a specific customer or the INV-ITEM records for a specific invoice by using the FOR parameter. See the FOR parameter for examples.

Selecting Subrecords for Listing (Subrecord-name)**Purpose**

The subrecord-name specification is the same as record-name, except that you use it to list the contents of subrecords. To display a subrecord, you must first open the subfile that contains the record.

Opening subfile

Remember that PRODUCT is a subrecord in the inventory file. To display the PRODUCT subrecords, you must first open the STOCK subfile. Because STOCK contains only one type of record or subrecord, you get the same result whether you enter the subrecord name or enter an unqualified LIST command:

```
open stock list product
```

or

```
open stock list
```


SCREEN
7-45

open stock list product

| STOCK-NO | CATEGORY | DESCRIPTION | PRICE | ΔMORE LIST□ |
|-------------|----------|------------------|--------|-------------|
| QTY-ON-HAND | | | | |
| . 101110 | SAW | TABLE 9IN 1.6HP | 189.88 | |
| | 24 | | | |
| . 101115 | SAW | TABLE 10IN 2.5HP | 299.95 | |
| | 18 | | | |
| . 101120 | SAW | TABLE 12IN 3.5HP | 499.95 | |
| | 21 | | | |
| . 102155 | SAW | CHAIN 18IN 3.7CU | 237.99 | |
| | 10 | | | |
| . 102160 | SAW | CHAIN 16IN 2.5CU | 207.99 | |
| | 13 | | | |
| . 102165 | SAW | CHAIN 14IN 2.0CU | 167.99 | |
| | 15 | | | |
| . 102170 | SAW | CHAIN 10IN 2.0CU | 77.99 | |
| | 14 | | | |
| . 103010 | SAW | CIRCULAR 6IN | 25.50 | |
| | 28 | | | |
| . 103013 | SAW | CIRCULAR 7IN | 34.50 | |
| | 31 | | | |
| . 103016 | SAW | CIRCULAR 8IN | 48.50 | |
| | 26 | | | |

7.19. SELECTING RECORDS THAT MEET A CONDITION (IF CLAUSE)

Function

The IF clause restricts the output of the LIST command on the basis of a conditional expression. You can use a conditional expression with or without the display-content-specification parameter. The format is:

Format

LIST [display-content-specification]

IF conditional-expression ...

Simple conditional expression

A simple conditional expression is a comparison between a record item and another record item or a literal value. The comparison operators are:

Comparison operators

EQ or = Equal to

NE Not equal to

GT or > Greater than

GE Greater than or equal to

LT or < Less than

LE Less than or equal to

LIST COMMAND*Comparing two record items*

Here's an example of a simple conditional expression comparing two record items in the inventory file. This example requests a display of the DESCRIPTION, ORDER-QTY, and COST items for all records in which the value of QTY-ON-HAND is less than the value of REORDER-POINT:

SCREEN
7-46

```
list description order-qty cost if qty-on-hand < reorder-point
```

| STOCK-NO | DESCRIPTION | ORDER-QTY | COST |
|----------|-------------------|-----------|--------|
| . 101110 | TABLE 9IN 1.6HP | 35 | 139.00 |
| . 101115 | TABLE 10IN 2.5HP | 30 | 223.00 |
| . 102170 | CHAIN 10IN 2.0HP | 20 | 50.00 |
| . 116550 | HAND 1/2IN 7/8HP | 10 | 44.50 |
| . 252777 | EXTENSION 36FT | 25 | 139.99 |
| . 252778 | EXTENSION 40FT | 25 | 159.99 |
| . 302295 | EDGER GAS | 25 | 110.00 |
| . 303377 | TRIMMER ELEC 16IN | 40 | 29.99 |
| . 304995 | RAKE 22IN SPREAD | 65 | 3.50 |
| . 567801 | SNAP-RING | 170 | 4.80 |
| . 567802 | SOLID-JOINT 6IN | 225 | 3.70 |
| . 567803 | SOLID-JOINT 8IN | 225 | 3.95 |
| . 567804 | ARC-JOINT 12IN | 130 | 4.65 |
| . 567805 | ARC-JOINT 16IN | 100 | 7.95 |
| . 571120 | 18-PC SET | 100 | 20.00 |
| . 571121 | 14-PC SET | 100 | 14.75 |
| . 571122 | 8-PC SET | 200 | 8.05 |
| . 572555 | SLOT 4IN | 600 | 1.25 |
| . 572556 | SLOT 6IN | 800 | 1.59 |

If you entered the same LIST command without the item names:

```
list if qty-on-hand < reorder-point
```

UNIQUE would display the entire contents of each record, meeting the condition you specified.

Comparing item with literal

Here's another example of a simple conditional expression, this time comparing an item in the record - CATEGORY - with the literal value LAWN:

SCREEN
7-47

```
list category description qty-on-hand if category = lawn
END LIST
```

| *STOCK-NO | CATEGORY | DESCRIPTION | QTY-ON-HAND |
|-----------|----------|-------------------|-------------|
| .301100 | LAWN | MOWER REAR BAG | 9 |
| .301110 | LAWN | MOWER ELEC W/CORD | 13 |
| .301120 | LAWN | MOWER BATTERY | 3 |
| .301130 | LAWN | MOWER SIDE BAG | 11 |
| .302295 | LAWN | EDGER GAS | 15 |
| .302296 | LAWN | EDGER ELECTRIC | 18 |
| .303377 | LAWN | TRIMMER ELEC 16IN | 20 |
| .303378 | LAWN | TRIMMER ELEC 10IN | 32 |
| .303379 | LAWN | TRIMMER ELEC 9IN | 45 |
| .304995 | LAWN | RAKE 22IN SPREAD | 32 |
| .304996 | LAWN | RAKE ADJUSTABLE | 12 |

*Literal smaller than
item*

When you compare an item to a nonnumeric literal value, the literal can have a smaller number of characters than the item actually contains. For instance, you might want to list the complete records for all the circular saws. In this example, we use EQ instead of the = symbol:

SCREEN
7-48

```
list if description eq circular
END LIST
```

| * STOCK-NO | CATEGORY | DESCRIPTION | QTY-ON-HAND |
|------------|----------|--------------|-------------|
| .103010 | SAW | CIRCULAR 6IN | 28 |
| 16.50 | 25.50 | 15 25 | 9 140 E410 |
| 51 | | | |
| .103013 | SAW | CIRCULAR 7IN | 31 |
| 22.50 | 34.50 | 25 45 | 10 90 E425 |
| 99 | | | |
| .103016 | SAW | CIRCULAR 8IN | 26 |
| 34.50 | 48.50 | 25 45 | 11 90 E430 |
| 107 | | | |

*Complex conditional
expression*

In addition to simple conditional expressions, you can specify more complex conditions for record selection. You can:

*NOT, AND, OR
operators*

- Precede a comparison operator with the word NOT to make it negative
- Use AND or OR to create a compound conditional expression
- Use combinations of NOT, AND, and OR. When you use a combination, UNIQUE evaluates NOT first, then AND, and finally OR. You can change the order of evaluation by using parentheses.

LIST COMMAND*Forming value range*

You can use AND and OR in more than one way. Our first example compares the same item name - PRICE - with two different literals. The effect of this expression is to form a value range:

SCREEN
7-49

```
list category description price if price gt 75 and < 100
```

| STOCK-NO | CATEGORY | DESCRIPTION | PRICE |
|----------|----------|--------------------|-------|
| 102170 | SAW | CHAIN 10IN 2.0CU | 77.99 |
| 121020 | TOOLBOX | CHEST 5 DRAWER | 99.99 |
| 121030 | TOOLBOX | CHEST 3 DRAWER | 94.99 |
| 251103 | LADDER | STEP 10FT | 84.99 |
| 267771 | SUMPPUMP | VERTICAL-STAINLESS | 84.95 |
| 268080 | SUMPPUMP | SUBMERSIBLE | 89.95 |
| 974452 | SANDER | BELT 2HP | 98.50 |
| 974453 | SANDER | BELT 1 | 83.50 |

Requiring two conditions

The next example is a combination of two separate comparisons, involving two item names and two literals. Only one record meets both of the conditions in this expression:

SCREEN
7-50

```
list category description cost if category = toolbox and cost gt 100
```

| STOCK-NO | CATEGORY | DESCRIPTION | COST |
|----------|----------|-----------------|--------|
| 121010 | TOOLBOX | CHEST 10 DRAWER | 105.99 |

7.20. LISTING A SUBSET OF A HIERARCHICAL FILE (FOR CLAUSE)

Function

The FOR clause lists items or complete records related to a particular record in a hierarchical file. The most common use of the FOR clause is to list child records for a specific parent record. The format is:

Format

```
LIST [[display-content-specification]
```

```
  [IF conditional-expression]:]... FOR identifier-1
```

We'll use the SALES file for the examples in this subsection. Remember, the SALES file contains three record types: CUSTOMER, INVOICE, and INV-ITEM.

Listing child records

Suppose you want to list all the invoices for ABLE COMPANY:

SCREEN 7-51

```
open sales
list invoice for 'able company'
```

END LIST

| ABLE COMPANY | | | | |
|--------------|----------|-------------|---------|-----------|
| INV-NO | INV-DATE | TOTAL-SALES | TAX-AMT | INV-GROSS |
| -357005 | 08/15/81 | 2,599.84 | 174.80 | 2,774.64 |
| -444834 | 09/10/81 | 580.17 | 28.30 | 608.47 |
| -455610 | 10/01/81 | 1,367.70 | 65.00 | 1,432.70 |

Perhaps now you want to list the items on invoice 357005:

SCREEN 7-52

```
list inv-item for 'able company',357005
```

END LIST

| ABLE COMPANY | | | | | | |
|--------------|-----------|--------------------|------|------------|-----------|--|
| -357005 | | | | | | |
| NMBR | ITEM-CODE | DESCRIPTION | QUAN | UNIT-PRICE | TOT-PRICE | |
| --01 | 121010 | TOOLBOX-CHEST 10DR | 6 | 144.99 | 869.94 | |
| --02 | 103010 | SAW-CIRCULAR 6IN | 12 | 25.50 | 306.00 | |
| --03 | 102165 | SAW-CHAIN 14IN | 4 | 167.99 | 671.96 | |
| --04 | 115000 | DRILL-PRESS 72IN | 1 | 340.94 | 340.94 | |
| --05 | 116550 | DRILL-HAND 7/8HP | 6 | 68.50 | 411.00 | |

Listing two record types

To list both the invoices and invoice items for Able Company, you can enter:

```
list invoice;inv-item for 'able company'
```

Listing all child records

Because you want to list all the records that are subordinate to Able Company, you can omit the record names and enter:

```
list for 'able company'
```

LIST COMMAND

In either case, you get the following listing:

SCREEN
7-53

```
list for 'able company'
```

| ABLE COMPANY | | | | | | | END LIST |
|--------------|-----------|--------------------|---------|------------|-----------|--|----------|
| * -INV-NO | INV-DATE | TOTAL-SALES | TAX-AMT | INV-GROSS | | | |
| * --NMBR | ITEM-CODE | DESCRIPTION | QUAN | UNIT-PRICE | TOT-PRICE | | |
| . -357005 | 08/15/81 | 2,599.84 | 174.80 | 2,774.64 | | | |
| . --01 | 121010 | TOOLBOX-CHEST 10DR | 6 | 144.99 | 869.94 | | |
| . --02 | 103010 | SAW-CIRCULAR 6IN | 12 | 25.50 | 306.00 | | |
| . --03 | 102165 | SAW-CHAIN 14IN | 4 | 167.99 | 671.96 | | |
| . --04 | 115000 | DRILL-PRESS 72IN | 1 | 340.94 | 340.94 | | |
| . --05 | 116550 | DRILL-HAND 7/8HP | 6 | 68.50 | 411.00 | | |
| . -444834 | 09/10/81 | 580.17 | 28.30 | 608.47 | | | |
| . --01 | 301110 | LAWN-MOWER ELEC | 3 | 193.39 | 580.17 | | |
| . -455610 | 10/01/81 | 1,367.70 | 65.00 | 1,432.70 | | | |
| . --01 | 101110 | SAW-TABLE 9IN | 2 | 189.88 | 379.76 | | |
| . --02 | 103016 | SAW-CIRCULAR 8IN | 8 | 48.50 | 388.00 | | |
| . --03 | 121020 | TOOLBOX-CHEST 5DR | 6 | 99.99 | 599.94 | | |

Listing specific items

Instead of listing complete records in the previous examples, you might want to list only certain items in the records. For example, suppose you want to list all the invoice numbers, dates, item numbers, codes, and quantities ordered for Able Company. You can omit the invoice and item numbers because they are record identifiers and are always listed when you request items in those records:

SCREEN
7-54

```
list inv.date item.code quan for 'able company'
```

| ABLE COMPANY | | | | END LIST |
|--------------|-----------|------|--|----------|
| * -INV-NO | INV-DATE | | | |
| * --NMBR | ITEM-CODE | QUAN | | |
| . -357005 | 08/15/81 | | | |
| . --01 | 121010 | 6 | | |
| . --02 | 103010 | 12 | | |
| . --03 | 102165 | 4 | | |
| . --04 | 115000 | 1 | | |
| . --05 | 116550 | 6 | | |
| . -444834 | 09/10/81 | | | |
| . --01 | 301110 | 3 | | |
| . -455610 | 10/01/81 | | | |
| . --01 | 101110 | 2 | | |
| . --02 | 103016 | 8 | | |
| . --03 | 121020 | 6 | | |

Using conditional expression

You can use a conditional expression in combination with your request for a child record listing. For example, suppose you want to list all invoices for ABLE COMPANY with dates earlier than 9/1/81:

```

SCREEN
7-55

list invoice if inv-date < 09/01/81 for 'able company'

                                END LIST

ABLE COMPANY
* -INV-NO  INV-DATE  TOTAL-SALES  TAX-AMT  INV-GROSS
. -357005  08/15/81    2,599.84    174.80   2,774.64

```

Other uses

The FOR clause has additional uses besides listing child records, or parts of child records, for a parent record. Two of these uses are illustrated in section 3:

1. When several parent records have the same child record (as in Figure 3-7), you can use the FOR clause to list parent records, or parts of parent records, that have the same child record. In that example, the parent record is an employee record and the child record gives an employee location. Because many employees have the same location, you can list all the employees for a particular location.
2. When a defined record has multiple identifiers, you can use the FOR clause to list records, or parts of records, that have the same major identifier. See Figure 3-20c for an example.

7.21. SPECIFYING A STARTING POINT FOR SELECTING RECORDS (AFTER/FROM CLAUSE)

Function

The AFTER/FROM clause designates a starting point from which you want to start listing records. The format is:

Format

```
LIST[[display-content-specification]
      [IF conditional-expression]:]... [FOR identifier-1]
```

```
{ AFTER } identifier-2
{ FROM  }
```

AFTER clause
FROM clause

When you use AFTER, the output listing starts with the record following the one you name with identifier-2. When you use FROM, the listing starts with the record you name with identifier-2.

LIST COMMAND*Listing specific items*

The following example, using INVFILE, requests a listing of the CATEGORY and QTY-ON-HAND items for all records starting at 260000. Notice in the output listing that there is no record with an identifier of 260000. Identifier-2 does not have to identify an actual record in the file – just a starting point:

SCREEN 7-56

```
list category qty-on-hand from 260000
```

| STOCK-NO | CATEGORY | QTY-ON-HAND | ΔMORE | LIST |
|----------|-------------|-------------|-------|------|
| . 267770 | SUMPPUMP | 8 | | |
| . 267771 | SUMPPUMP | 6 | | |
| . 268080 | SUMPPUMP | 2 | | |
| . 269000 | SUMPPUMP | 2 | | |
| . 301100 | LAWN | 9 | | |
| . 301110 | LAWN | 13 | | |
| . 301120 | LAWN | 3 | | |
| . 301130 | LAWN | 11 | | |
| . 302295 | LAWN | 15 | | |
| . 302296 | LAWN | 18 | | |
| . 303377 | LAWN | 20 | | |
| . 303378 | LAWN | 32 | | |
| . 303379 | LAWN | 45 | | |
| . 304995 | LAWN | 32 | | |
| . 304996 | LAWN | 12 | | |
| . 315555 | WHEELBARROW | 16 | | |
| . 315556 | WHEELBARROW | 12 | | |
| . 316111 | WHEELBARROW | 23 | | |
| . 317995 | WHEELBARROW | 11 | | |

Listing whole records

If you omitted the display content specification:

```
list from 260000
```

UNIQUE would list the complete contents of all records from the starting point you specified.

Using with IF and FOR clauses

You can also use the AFTER/FROM clause in combination with the IF and FOR clauses. We'll illustrate only the IF clause:

SCREEN 7-57

```
list category qty-on-hand if qty-on-hand LE 3 after 260000
```

| STOCK-NO | CATEGORY | QTY-ON-HAND | END | LIST |
|----------|----------|-------------|-----|------|
| . 268080 | SUMPPUMP | 2 | | |
| . 269000 | SUMPPUMP | 2 | | |
| . 301120 | LAWN | 3 | | |

7.22. GENERATING STATISTICS (STATISTICAL-FUNCTION)

Function The final parameter on the LIST command generates statistics about the records you select with the other LIST parameters. The format is:

Format

```
LIST [[display-content-specification]
      [IF conditional-expression];]...
      [FOR identifier-1] { [AFTER] identifier-2
                        [FROM]
                        }
      statistical-function [item-name-1[, item-name-2]...]...
```

Statistical functions where:

AVG item-name-1[, item-name-2]...
Displays the average value for each item you specify.

COUNT
Displays the number of records that meet the selection criteria you establish with the other LIST command parameters.

MAX item-name-1[, item-name-2]...
Displays the maximum value for each item you specify and the identifier of the record that contains it.

MIN item-name-1[, item-name-2]...
Displays the minimum value for each item you specify and the identifier of the record that contains it.

TOTAL item-name-1[, item-name-2]...
Displays the total value for each item you specify.

Item-names Item-names are the names that appear as column headers, except for the identifier. Any items you name in statistical functions must be defined as numeric in the data definition.

LIST COMMAND

Example using all functions

The following example requests all five statistical functions for records in INVFILE whose price is greater than 75 and less than 100. You can specify statistical functions in any order. The semicolon before the first statistical function is optional:

SCREEN 7-58

```
list category description price if price gt 75 and < 100: count
avg price max price min price total price
```

| STOCK-NO | CATEGORY | DESCRIPTION | PRICE |
|----------|----------|--------------------|-------|
| 102170 | SAW | CHAIN 18IN 2.0CU | 77.99 |
| 121020 | TOOLBOX | CHEST 5 DRAWER | 99.99 |
| 121030 | TOOLBOX | CHEST 3 DRAWER | 94.99 |
| 251103 | LADDER | STEP 10FT | 84.99 |
| 267771 | SUMPPUMP | VERTICAL-STAINLESS | 84.95 |
| 268080 | SUMPPUMP | SUBMERSIBLE | 89.95 |
| 974452 | SANDER | BELT 2HP | 98.50 |
| 974453 | SANDER | BELT 1 | 83.50 |

COUNT STOCK-NO= 8 TOTAL PRICE= 714.86 AVG PRICE= 89.358
MIN PRICE= 77.99 FOR 102170 MAX PRICE= 99.99 FOR 121020

Hierarchical file

When you use statistical functions with a hierarchical file like SALES, UNIQUE compiles statistics not only for the records you select for listing, but also for any records below them in the hierarchy. For example, when you request a listing of invoices for Able Company, you get statistical data about both the invoices and the items on the invoices. Statistics about invoice items are displayed on detail lines following each invoice record, with a number sign (#) preceding each detail line. Cumulative statistics for Able Company are displayed on a total line, also preceded by a number sign:

SCREEN 7-59

```
open sales
list invoice for 'able company': total tot-price total-sales tax-amt.
inv-gross count
```

| ABLE COMPANY | INV-NO | INV-DATE | TOTAL-SALES | TAX-AMT | INV-GROSS |
|--------------|--------------|-----------------------------|---------------------------|---------------------------|-----------|
| | -357005 | 08/15/81 | 2,599.84 | 174.80 | 2,774.64 |
| # | -357005: | COUNT NMBR= 5 | TOTAL TOT-PRICE= 2,599.84 | | |
| | -444834 | 09/10/81 | 580.17 | 28.30 | 608.47 |
| # | -444834: | COUNT NMBR= 1 | TOTAL TOT-PRICE= 580.17 | | |
| | -455610 | 10/01/81 | 1,367.70 | 65.00 | 1,432.70 |
| # | -455610: | COUNT NMBR= 3 | TOTAL TOT-PRICE= 1,367.70 | | |
| # | ABLE COMPANY | COUNT INV-NO= 3 | COUNT NMBR= 9 | TOTAL TOT-PRICE= 4,547.71 | |
| | | TOTAL TOTAL-SALES= 4,547.71 | TOTAL TAX-AMT= 268.10 | | |
| | | TOTAL INV-GROSS= 4,815.81 | | | |

Because TOT-PRICE is an item in the INV-ITEM record, UNIQUE provides a total TOT-PRICE for each invoice and a grand total for the Able Company. For the COUNT function, UNIQUE lists the number of INV-ITEM records for each invoice (UNIQUE identifies the INV-ITEM records by their identifier, NMBR) and the grand total of INV-ITEM records for Able Company. UNIQUE also lists the number of INVOICE records for Able Company, identifying them as INV-NO.

MORE COMMAND**7.23. REQUESTING THE NEXT LIST OR DETAIL SCREEN**

Function The MORE command requests the next screenful of data from the previous LIST or DETAIL command. The format is:

Format MORE [{DETAIL}]
 [LIST]

Embedded MORE command As we showed in the LIST examples, UNIQUE includes the MORE command in the screen display for the LIST and DETAIL commands when the output listing takes up more than one screen. You simply press the TRANSMIT key (or CTRL/C on a hard-copy terminal) to display the next screenful of data.

Requesting next screen later Instead of displaying the next LIST or DETAIL screen immediately, you may want to enter another UNIQUE command. You can enter any UNIQUE command (except OPEN or CLOSE) between LIST or DETAIL output screens. You then resume processing the LIST or DETAIL command by entering the MORE command.

MORE without LIST or DETAIL When you enter the MORE command without the LIST or DETAIL option:

MORE

UNIQUE displays the next screenful from the most recent LIST or DETAIL command.

MORE with LIST or DETAIL If you have both LIST and DETAIL commands outstanding, enter:

more list

or

more detail

to tell UNIQUE which command to resume processing.

See the DETAIL command (7.24) for examples of the MORE command.

7.24. OBTAINING A SECONDARY LISTING

**Function**

The **DETAIL** command interrupts the processing of a **LIST** command to obtain a more detailed listing. Its format is the same as the **LIST** command except for the command name:

Format

```

DETAIL [ [display-content-specification] ] ...
      [ [IF conditional-expression]; ]
      [FOR identifier-1]
      [ {AFTER} identifier-2 ]
      [ {FROM } ]
      [statistical-function [item-name-1[, item-name-2]...] ] ...

```

LIST output saved

The **DETAIL** command functions exactly the same way as the **LIST** command. For a description of the format specifications, see the **LIST** command (7.16).

The **DETAIL** command lets you obtain a secondary listing without destroying the uncompleted output from the first listing. **UNIQUE** saves the text of the **DETAIL** command separately from that of the **LIST** command. This means that you can go back and forth between two different listing operations. You can also enter other **UNIQUE** commands while saving the data from both the **LIST** and **DETAIL** commands. The **MORE** command reinstates the **LIST** or **DETAIL** processing.

MORE command**Example**

The following example shows a sequence of **LIST**, **DETAIL**, and **MORE** commands. The **LIST** command requests a listing of the **CUSTOMER** records in the **SALES** file. The first **DETAIL** command requests a listing of the **INVOICE** records for Able Company, and the second **DETAIL** requests a listing of the **INV-ITEM** records for invoice 357005. Because the **DETAIL** output is complete, you can enter either **MORE** or **MORE LIST** to reinstate the **LIST** command:

DETAIL COMMAND

SCREEN
7-60

open sales list customer

| * CUST-NAME PHONE | CUST-ID | ADDRESS | CITY | △MORE LIST□ | |
|--------------------------------|---------|------------------|------------|-------------|-------|
| | | | | ST | ZIP |
| ABLE COMPANY 675-887-9595 | 5-40590 | 2046 CLOVERLY LA | MIDDLETOWN | PA | 19789 |
| ADDLEBY, INC. 543-129-1187 | 5-21543 | 1519 MEARNS RD | HOMETOWN | AR | 54331 |
| ADHOC, INC. 236-657-5446 | 5-13448 | 250 HOPEWELL | GREENVILLE | GA | 34687 |
| AMARK HARDWARE 625-537-9954 | 5-60814 | 29 CATANGA ST | CATANGA | FL | 42970 |
| AMWAY VARIETY 457-332-5648 | 5-15574 | 5348 MARIAN RD | SEKONDA | WI | 68867 |
| ARROWHEAD LTD. 145-054-6700 | 5-55429 | 8410 BROAD ST | BARRINGTON | MA | 30558 |
| ASTOR HARDWARE 323-776-4289 | 5-18976 | 129 MAIN ST | ASTOR | SC | 25995 |
| ATRIUM COMPANY 278-246-6785 | 5-05946 | 550 FORREST AVE | DRINKWATER | VA | 32584 |
| ATWELL VARIETY 817-236-5579 | 5-32963 | 1846 DARIEN DR | PHILA. | PA | 19950 |
| B & M HARDWARE 345-886-7896 | 5-40896 | 1226 SHADY LA | ROCKWELL | IN | 45680 |

SCREEN
7-61

detail invoice for 'able company'

END DETAIL

| ABLE COMPANY | | | | | |
|--------------|----------|-------------|---------|-----------|--|
| * -INV-NO | INV-DATE | TOTAL-SALES | TAX-AMT | INV-GROSS | |
| -357005 | 08/15/81 | 2,599.84 | 174.80 | 2,774.64 | |
| -444834 | 09/10/81 | 580.17 | 28.30 | 608.47 | |
| -455610 | 10/01/81 | 1,367.70 | 65.00 | 1,432.70 | |

SCREEN
7-62

detail inv-item for 'able company'.357005

END DETAIL

| ABLE COMPANY | | | | | | |
|--------------|-----------|--------------------|------|------------|-----------|--|
| -357005 | | | | | | |
| * --NMNR | ITEM-CODE | DESCRIPTION | QUAN | UNIT-PRICE | TOT-PRICE | |
| --01 | 121010 | TOOLBOX-CHEST 10DR | 6 | 144.99 | 869.94 | |
| --02 | 103010 | SAW-CIRCULAR 6IN | 12 | 25.50 | 306.00 | |
| --03 | 102165 | SAW-CHAIN 14IN | 4 | 167.99 | 671.96 | |
| --04 | 115000 | DRILL-PRESS 72IN | 1 | 340.94 | 340.94 | |
| --05 | 116550 | DRILL-HAND 7/8HP | 6 | 68.50 | 411.00 | |

SCREEN
7-63

more list

| * CUST-NAME PHONE | CUST-ID | ADDRESS | CITY | △MORE LIST□ | |
|-------------------------------------|---------|-------------------|-------------|-------------|-------|
| | | | | ST | ZIP |
| BAG 'N SAVE 356-957-6672 | 4-85794 | 385 GREENTREE ST | HAMILTON | MD | 56439 |
| BALTIC HOUSEWARES 467-786-6657 | 5-43875 | 6540 BALTIC AVE | CLAREMONT | DE | 35775 |
| BARBARA'S GREENERY 817-349-1865 | 4-58856 | 9690 GARDEN LANE | ST. COLLEGE | PA | 34680 |
| BARNEGAT LIGHT CO. 287-944-1376 | 4-50613 | 518 CANAL ST | BARNEGAT | MA | 25983 |
| BARRINGTON HARDWARE 689-212-1856 | 5-18900 | 1211 FITZWATER ST | BARRINGTON | NJ | 14960 |
| BARRY'S GARDEN MART 854-256-7171 | 4-85437 | 3775 ALLEN RD | ALLENTOWN | PA | 18494 |
| BATTLESON HARDWARE 785-556-2381 | 5-57556 | 6885 BATTLESON ST | COMMODORE | KY | 66645 |
| BELMONT CYCLE 355-045-7883 | 4-05854 | 3590 PARKSIDE AV | ABINGTON | PA | 52960 |
| BENSON LAWN CARE 256-657-7556 | 4-48676 | 534 JACKSON AV | WARMINSTER | RI | 42955 |
| BENTLEY'S VARIETY | 4-65944 | 8180 VICTORY DR | HAMILTON | VT | 62586 |

SHOW COMMAND

7.25. DISPLAYING RECORD FORMATS AND COMMAND STATUS



Function The SHOW command displays:

- the format of each record and subrecord in the defined file with update formats for the items in the record;
- the most recent LIST and DISPLAY commands; and
- any DISPLAY, DELETE, or display format ADD or CHANGE command with unprocessed identifiers (requiring a NEXT command for processing).

The format of the SHOW command is:

Format SHOW

Record format display In the record format display, UNIQUE gives the name of each record and its identifier and item names. Below the identifier and item names, UNIQUE displays update formats, using special symbols to show whether a value is required in an ADD command and whether the value may be changed in a CHANGE command. These symbols and their meanings are listed in Table 7-1. The number of symbols in each field indicates the length of the field.

Update formats

Table 7-1. Symbols Used in SHOW Command Update Formats

| Symbol | Meaning | ADD Command | CHANGE Command |
|--------|--------------|--------------------|----------------------|
| I | Identifier | Value required | Change not permitted |
| A | | Value required | Change not permitted |
| D | Display only | Value not required | Change not permitted |
| ! | | Value required | Change permitted |
| ✱ | | Value not required | Change permitted |

SHOW COMMAND

```

SCREEN
7-65

open invfile show

                                END  SHOW
                                INVFILE
* I-REC  STOCK-NO  CATEGORY  DESCRIPTION  QTY-ON-HAND
      !!!!!  AAAAAAAAAA  *****
      COST  PRICE  REORDER-POINT  ORDER-QTY  KILOS  GRAMS  LOC
      !!!!!  !!!!!  !!!!!  !!!!!  !!!!!  !!!!!  !!!!!
      YTD-USAGE
      DD,DDD
* PRODUCT  STOCK-NO  CATEGORY  DESCRIPTION  PRICE
      !!!!!  DDDDDDDDDDD  DDDDDDDDDDDDDDDDD  D.DDD.DD
      QTY-ON-HAND
      DD,DDD

LIST CATEGORY DESCRIPTION PRICE IF PRICE GT 75 AND < 100; COUNT AVG PRICE
MAX PRICE MIN PRICE TOTAL PRICE

```

In this example, the YTD-USAGE item in I-REC is for display only. A value is not required in order to add a record, and change is not permitted. All of the items in the subrecord PRODUCT are for display only, because no updating of PRODUCT is permitted in the data definition.

The only command shown in the example is LIST, because no DETAIL command was issued for INVFILE and no DISPLAY, DELETE, ADD, or CHANGE commands are active.

You can also display the format of the PRODUCT record by entering a SHOW command for the STOCK subfile:

```

SCREEN
7-66

open stock show

                                END  SHOW
                                STOCK
* PRODUCT  STOCK-NO  CATEGORY  DESCRIPTION  PRICE
      !!!!!  DDDDDDDDDDD  DDDDDDDDDDDDDDDDD  D.DDD.DD
      QTY-ON-HAND
      DD,DDD

LIST PRODUCT

```

PART 4. APPENDIXES



Appendix A. Format Presentation and Coding Rules

A.1. GENERAL RULES FOR THIS DOCUMENT

The following rules apply to the presentation of formats in this document, the coding of your data definition, and use of UNIQUE.

*Capital letters,
punctuation marks*

- Code or enter capital letters, punctuation marks (except braces, brackets, and ellipses) exactly as shown. For example, code:

ALLOW CHANGE

as:

ALLOW CHANGE

Enter:

OPEN password

as:

OPEN SALES

Lowercase terms

- You must supply all data for lowercase terms. For example, enter:

PARENT IS defined-record-name-2

as:

PARENT IS EMPLOYEE

Braces

- For data within braces { }, you must choose one of the entries. For example, code:

```
{FROM stored-record-name-1
 {FROM REPEATING GROUP data-name-1}
```

as:

```
FROM EMPLOYEE-REC
```

Brackets

- For data within brackets [], including commas or semicolons, use or omit entries according to program requirements. If you include an entry within brackets, you must also choose one of the entries within braces. For example, code:

```
[ALLOW {ADD
 {DELETE
 {ADD AND DELETE} } OF RECORD]
```

as:

```
ALLOW ADD OF RECORD
```

Ellipsis

- An ellipsis means a variable number of entries are present. For example, enter:

```
DISPLAY identifier-1 [, identifier-2]...
```

as:

```
DISPLAY 'CENTURY HARDWARE', 'BARRY'S GARDEN MART'
```

**Lowercase and
reverse print**

- In screen displays, lowercase and reverse print represent entries that you make.

Literals

- Enclose alphabetic and alphanumeric literals in single quotes to distinguish them from your data names. Do not enclose numeric literals in quotes, except in the PREFIX and FILL KEY clauses.

Coding form

- Use the standard COBOL coding form to code your data definition. As noted in their descriptions, certain statements begin in margin A (column 8) of the coding form. Begin all other statements in column 12 or beyond, but do not code past column 72.

*Periods,
semicolons*

- Follow statements in the identification and data divisions with periods. Use semicolons at your option throughout the defined file definition; they are ignored by the data definition processor.

Appendix B. Reserved Words

Table B-1 lists reserved words you must not use for terms that you supply in the definition division of the data definition.

Except for the words DEFINITION and DIVISION, you can use these reserved words in the data division. The COBOL reserved word list shown in Table B-2 applies to the data division of the data definition.

Table B-1. Reserved Words in the Definition Division

| | | | |
|-------------|------------|-------------------------|------------|
| ADD | DBS | KEY-NAME | SELECTIVE |
| ALL | DEFINED | MANUAL | SEMICOLON |
| ALLOW | DEFINITION | MUST | SET |
| ALSO | DELETE | NEUTRAL | SUBFILE |
| AND | DIVISION | NEXT-MEMBER- POINTER | SUBRECORD |
| ARE | DMS | OF | SUPPLEMENT |
| AS | DUPLICATE | ONLY | THROUGH |
| ASSUME | FILE | OWNED | THRU |
| ASSUMES | FILL | OWNING | TO |
| BREAK | FOLLOWS | PARENT | TOTAL |
| BY | FROM | PASSWORD | TYPE |
| CALC | GROUP | PERIOD | UPDATE |
| CHANGE | HIDDEN | POINTER | USING |
| CONTAINS | IDENTIFIER | PREFIX | VALUE |
| CONTROL | IN | RECORD | VALUES |
| CONTROLLED | IS | REPEATING | VIA |
| CONTROLLING | ITEM | ROLE | WITHIN |
| COUNT | KEY | | |

Table B-2. COBOL Reserved Words in the Data Division (Part 1 of 2)

| | | |
|--------------------|--------------------|-----------------|
| ACCEPT | DATE-COMPILED | INSTALLATION |
| ACCESS | DATE-WRITTEN | INTO |
| ACTUAL | DECIMAL-POINT | INVALID |
| ADD | DECLARATIVES | IS |
| ADVANCING | DEPENDING | JUST |
| AFTER | DESCENDING | JUSTIFIED |
| ALL | DIRECT | KEY |
| ALPHABETIC | DISC | LABEL |
| ALTER | DISC-n | LEADING |
| ALTERNATE | DISPLAY | LEFT |
| AND | DIVIDE | LESS |
| APPLY | DIVISION | LINE |
| ARE | DOWN | LINES |
| AREA | EBCDIC | LINKAGE |
| AREAS | ELSE | LOCK |
| ASCENDING | END | LOW-VALUE |
| ASCII | ENDING | LOW-VALUES |
| ASSIGN | ENTER | MAP |
| AT | ENTRY | MASTER-INDEX |
| AUTHOR | ENVIRONMENT | MEMORY |
| BEFORE | EQUAL | MODE |
| BEGINNING | EQUALS | MODULES |
| BLANK | ERROR | MONITOR |
| BLOCK | EVERY | MORE-LABELS |
| BLOCK-COUNT | EXAMINE | MOVE |
| BLOCK-LENGTH-CHECK | EXCEEDS | MULTIPLE |
| BUFFER-OFFSET | EXHIBIT | MULTIPLY |
| BY | EXIT | NAMED |
| CALL | EXTENDED | NEGATIVE |
| CARD-PUNCH | EXTENDED-INSERTION | NEXT |
| CARD-READER | FD | NO |
| CARD-READER-51 | FILE | NOT |
| CARD-READER-66 | FILE-CONTROL | NOTE |
| CHARACTER | FILE-LIMIT | NUMERIC |
| CHARACTERS | FILE-LIMITS | OBJECT-COMPUTER |
| CHANGED | FILE-PREPARATION | OCCURS |
| CLOSE | FILLER | OF |
| COBOL | FIRST | OFF |
| COMMA | FOR | OMITTED |
| COMP | FORM-OVERFLOW | ON |
| COMP-1 | FROM | OPEN |
| COMP-2 | GENERATE | OPTIONAL |
| COMP-3 | GIVING | OR |
| COMP-4 | GO | ORGANIZATION |
| COMPUTATIONAL | GREATER | OTHERWISE |
| COMPUTATIONAL-1 | HIGH-VALUE | OUK-n |
| COMPUTATIONAL-2 | HIGH-VALUES | OUTPUT |
| COMPUTATIONAL-3 | I-O | PERCENT |
| COMPUTATIONAL-4 | I-O-CONTROL | PERFORM |
| COMPUTE | IDENTIFICATION | PIC |
| CONFIGURATION | IF | PICTURE |
| CONTAINS | IN | POSITION |
| COPY | INDEX | POSITIVE |
| CORR | INDEXED | PRINTER |
| CORRESPONDING | INDICES | PROCEDURE |
| CURRENCY | INITIATE | PROCEED |
| CYLINDER-INDEX | INPUT | PROCESSING |
| CYLINDER-OVERFLOW | INPUT-OUTPUT | PROGRAM |
| DATA | INSERT | PROGRAM-ID |

Table B-2. COBOL Reserved Words in the Data Division (Part 2 of 2)

| | | |
|---------------|-----------------|-----------------|
| QUOTE | SENTENCE | TAPE-6 |
| QUOTES | SEPARATE | TAPES |
| RANDOM | SEQUENTIAL | TERMINATE |
| READ | SET | THAN |
| READY | SIGN | THEN |
| RECORD | SIZE | THROUGH |
| RECORDING | SORT | THRU |
| RECORDS | SOURCE-COMPUTER | TIME |
| REDEFINES | SPACE | TIMES |
| REEL | SPACES | TO |
| RELATIVE | SPECIAL-NAMES | TRACE |
| RELEASE | STANDARD | TRACKS |
| REMAINDER | STATUS | TRAILING |
| REMARKS | STOP | TRANSFORM |
| RENAMES | SUBTRACT | UNEQUAL |
| REPLACING | SYMBOLIC | UNIT |
| RERUN | SYNC | UNIVAC-n |
| RESERVE | SYNCHRONIZED | UNTIL |
| RESET | SYSCHAN-n | UP |
| RESTRICTED | SYSCOM | UPON |
| RETURN | SYSCONSOLE | USAGE |
| REVERSED | SYSDATE | USE |
| REWIND | SYSERR | USING |
| REWRITE | SYSERR-n | VALUE |
| RIGHT | SYSIN | VALUES |
| ROUNDED | SYSIN-96 | VARYING |
| RUN | SYSIN-128 | VERIFY |
| SAME | SYSLOG | WHEN |
| SD | SYSLST | WITH |
| SEARCH | SYSSWCH | WORDS |
| SECTION | SYSSWCH-n | WORKING-STORAGE |
| SECURITY | SYSTIME | WRITE |
| SEEK | TALLY | ZERO |
| SEGMENT-LIMIT | TALLYING | ZEROES |
| SELECT | TAPE | ZEROS |



Appendix C. Data Definition Processor Diagnostics

The error diagnostics issued by the data definition processor are shown in Table C-1.

Table C-1. Compilation Time Diagnostics Unique to the IMS Data Definition Processor (Part 1 of 2)

| Message Number | Severity Code | Diagnostic Message | Explanation | | |
|----------------|---------------|---|---|--|---|
| | | | Reason | Rule | Recovery |
| 139 | U | —SUSPEND CHECKING INVALID SOURCE STATEMENTS ON THIS LINE. | Beginning at this source line, the data definition processor does not recognize source input as data definition language. | The processor does no validity checking for syntax of data definition source statements until it recognizes some succeeding statement. | If preceded by another diagnostic for the same line number, you usually have to recover for that diagnostic only, but the remainder of this line <i>might</i> contain another error. If no other diagnostic precedes it, this line contains an error that's embedded in an unrecognized statement type. |
| 140 | U | —RESUME CHECKING SOURCE STATEMENTS ON THIS LINE. | After issuing diagnostic 139, the data definition processor again recognizes source input as data definition language. | Error processing continues, beginning with this source line. | None required. Before recompiling, scan all lines skipped in validity checking for <i>possible</i> error. |
| 159 | U | REFERENCE TO insert INVALID | Self-explanatory | Refer to Appendixes A and B for the rules for each statement. | Correct the data definition and recompile. |
| 160 | U | DEFINITION IS TOO LARGE | The length of the data definition record exceeds the block size specified for the NAMEREC file. | Block size for the NAMEREC file specified with the NBLK keyword parameter of the IMSCONF jproc or the BLKSZE parameter of the ZP#NRU utility. It ranges from 1024 to 12,800 bytes but must not exceed disk track size. | Reduce size of the data definition record and recompile. The indicated line number caused the overflow. Or, specify a larger block size for the NAMEREC file and reconfigure. |

Table C-1. Compilation Time Diagnostics Unique to the IMS Data Definition Processor (Part 2 of 2)

| Message Number | Severity Code | Diagnostic Message | Explanation | | |
|----------------|---------------|--|--|---|--|
| | | | Reason | Rule | Recovery |
| 161 | C | CHANGE TO NEUTRAL SUPPLEMENT IS ILLEGAL. | The processor has encountered the ALLOW CHANGE option specified for an item in a supplement that has no ROLE IN UPDATE specified or whose ROLE IN UPDATE is NEUTRAL. | If you specify the ALLOW CHANGE option for an item, the supplement's ROLE IN UPDATE must be CONTROLLED. | Correct the data definition and recompile. You may also need to revise your action program logic. |
| 162 | C | CHANGE TO CONTROLLING SUPPLEMENT IS ILLEGAL. | The processor has encountered the ALLOW CHANGE option specified for an item in a supplement whose ROLE IN UPDATE is CONTROLLING. | Same as 161. | Same as 161. |
| 163 | C | ADD TO NEUTRAL SUPPLEMENT IS ILLEGAL. | The processor has encountered a MUST ADD option specified for an item in a supplement that has no update role specified or whose ROLE IN UPDATE is NEUTRAL. | If you specify the MUST ADD option for an item, the supplement's ROLE IN UPDATE must be CONTROLLED. | Same as 161. |
| 164 | C | ADD TO CONTROLLING SUPPLEMENT IS ILLEGAL. | The processor has encountered a MUST ADD option specified for an item in a supplement whose ROLE IN UPDATE is CONTROLLING. | Same as 163. | Same as 161. |
| 165 | U | CANNOT ADD OR DELETE CONTROL BREAK. | The processor has encountered an ALLOW ADD, ALLOW DELETE, or ALLOW ADD AND DELETE clause specified for a defined record with a FROM CONTROL BREAK clause specified. | You cannot specify the ALLOW ADD AND DELETE clause for a defined record with FROM CONTROL BREAK specified. | Same as 161. |
| 166 | U | CANNOT ADD OR DELETE REPEATING GROUP. | The processor has encountered an ALLOW ADD, ALLOW DELETE, or ALLOW ADD AND DELETE clause specified for a defined record with a FROM REPEATING GROUP clause specified. | You cannot specify the ALLOW ADD AND DELETE clause for a defined record with FROM REPEATING GROUP specified. | Same as 161. |
| 167 | U | SEE CONSOLE FOR DMXX | OS/3 has issued a numbered data management error message to the system console, reflecting an error detected during processing of the NAMEREC file by the data definition processor. | None. The actual OS/3 data management message number, the prefix of which is "DM", appears at the system console and on the console output printer (COP) sheet for the run. | According to the nature of the error detected or reported to data management. Refer to the OS/3 system messages operator/programmer reference. |

Appendix D. UNIQUE Lexicon

Lexicon record

The IMS configurator generates a lexicon record when you specify UNIQUE=YES or RES in the OPTIONS section. The lexicon record consists of the language elements used in UNIQUE commands and responses.

Defining lexicons

You can define UNIQUE lexicons by including LANGUAGE sections in your configurator input. (See the IMS system support functions user guide, UP-8364 (current version).) When you omit the LANGUAGE section or specify OPEN as the lexicon-id, the configurator generates the standard UNIQUE lexicon.

Tables D-1 through D-6 list the language elements in the standard UNIQUE lexicon.

NOTE:

You cannot change symbolic terms used in UNIQUE commands. You can only change alphabetic terms.

Table D-1. UNIQUE Commands

| Command | Description |
|---------|--|
| OPEN | Begins a dialog |
| CLOSE | Terminates a dialog |
| DISPLAY | Displays a record in a file |
| DELETE | Deletes a record in a file |
| OK | Finalizes update operation |
| CANCEL | Cancels current update operation |
| ADD | Inserts a record into a file |
| CHANGE | Changes a record in a file |
| NEXT | Processes the next record, as mentioned in preceding command |
| LIST | Lists records from a file |
| MORE | Continues processing previous LIST or DETAIL command output |
| DETAIL | Performs secondary listing operation |
| SHOW | Displays information about current dialog |

Table D-2. Punctuation Used in UNIQUE Commands

| Symbol | Description |
|--------|---|
| ; | Semicolon. Separates identifiers and record names. |
| ' | Single quote or apostrophe. Delimits names that include special characters. |
| - | Hyphen. Replaces parent record identifier. |
| , | Comma. Separates parent/child identifiers. Also used in editing numbers. |
| (| Left parenthesis. Used in arithmetic expressions. |
|) | Right parenthesis. |

Table D-3. List Command Keywords

| Keyword | Description |
|---------|--|
| FOR | Restricts output from LIST command |
| FROM | Specifies records to be listed |
| AFTER | Specifies records to be listed |
| IF | Initiates selection criteria in LIST command |
| ALL | Specifies records to be listed |

Table D-4. List Command Logical Operators

| Operator | Description |
|----------|-----------------------------------|
| AND | Intersection operator |
| OR | Union operator |
| NOT | Negation operator |
| EQ or = | Equals operator |
| NE | Not equals operator |
| GT or > | Greater-than operator |
| GE | Greater-than-or-equal-to operator |
| LT or < | Less-than operator |
| LE | Less-than-or-equal-to operator |

Table D-5. List Command Statistical Functions

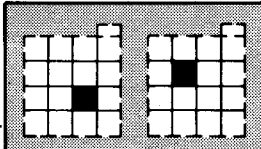
| Function | Description |
|----------|-------------------|
| TOTAL | Total operation |
| COUNT | Count operation |
| MAX | Maximum operation |
| MIN | Minimum operation |
| AVG | Average operation |

Table D-6. Words Used in UNIQUE Status and Error Messages

| | | | |
|----------|----------|----------|----------|
| ALLOWED | EXCEEDS | LESS | RELATION |
| ARGUMENT | EXISTS | LEXICON | REQUEST |
| ARITH. | EXPRESS. | LITERAL | RESPONSE |
| BAD | FEW | LOGICAL | SINGLE |
| BOUND | FILE | LONG | START |
| CANCELED | FOUND | LOWER | STATIST. |
| CHANGED | FUNCTION | MANY | SUCH |
| CLOSED | HAS | MESSAGE | THAN |
| COMMAND | IDENT. | MISSING | TOO |
| COMPLETE | ILLEGAL | NAME | TOO BIG |
| COMPOSED | IN | NO | UNDEFINE |
| DATA | INCOMPL. | NUMERIC | UNRECOG. |
| DATATYPE | INPUT | OPERAND | USAGE |
| DEFINED | INVALID | OPERATOR | VALUE |
| DIGIT | I/O | PAGE | VERB |
| END | ITEM | PAREN. | WORD |
| EOM | KNOWN | PASSWORD | ZERO |
| ERROR | LATE | RECORD | |

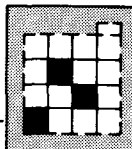
Appendix E. Data Definitions for UNIQUE Examples

Figures E-1 and E-2 show data definitions for the defined files used in the UNIQUE examples.



| LINE NO. | SEQ. | SOURCE STATEMENT |
|----------|------|---|
| 00001 | | IDENTIFICATION DIVISION. |
| 00002 | | PROGRAM-ID. UNIQUE-1. |
| 00003 | | DATA DIVISION. |
| 00004 | | FILE SECTION. |
| 00005 | | FD INVFILE. |
| 00006 | | 01 INVREC. |
| 00007 | | 02 STOCK-NO PIC X(6). |
| 00008 | | 02 CATEGORY PIC X(12). |
| 00009 | | 02 DESCRIPTION PIC X(18). |
| 00010 | | 02 QTY-ON-HAND PIC 9(5) USAGE COMP-3. |
| 00011 | | 02 COST PIC 9(6)V99 USAGE COMP-3. |
| 00012 | | 02 PRICE PIC 9(6)V99 USAGE COMP-3. |
| 00013 | | 02 REORDER-POINT PIC 9(5) USAGE COMP-3. |
| 00014 | | 02 ORDER-QTY PIC 9(5) USAGE COMP-3. |
| 00015 | | 02 WEIGHT. |
| 00016 | | 03 KILOS PIC 9(3). |
| 00017 | | 03 GRAMS PIC 9(3). |
| 00018 | | 02 LOCATION PIC X(4). |
| 00019 | | 02 YTD-USAGE PIC 9(5) USAGE COMP-3. |
| 00020 | | DEFINITION DIVISION. |
| 00021 | | DEFINED FILE INVFILE PASSWORD |
| 00022 | | DEFINED RECORD I-REC FROM INVREC |
| 00023 | | ALLOW ADD AND DELETE |
| 00024 | | IDENTIFIER STOCK-NO |
| 00025 | | ITEM CATEGORY MUST ADD |
| 00026 | | ITEM DESCRIPTION ALLOW CHANGE |
| 00027 | | ITEM QTY-ON-HAND |
| 00028 | | ITEM COST ALLOW CHANGE VALUE IS '00001.00' THRU '20000.00' |
| 00029 | | MUST ADD |
| 00030 | | ITEM PRICE ALLOW CHANGE VALUE IS '00001.00' THRU '30000.00' |
| 00031 | | MUST ADD |
| 00032 | | ITEM REORDER-POINT ALLOW CHANGE |
| 00033 | | ITEM ORDER-QTY ALLOW CHANGE |
| 00034 | | ITEM KILOS ALLOW CHANGE |
| 00035 | | ITEM GRAMS ALLOW CHANGE |
| 00036 | | ITEM LOC FROM LOCATION ALLOW CHANGE MUST ADD |
| 00037 | | ITEM YTD-USAGE |
| 00038 | | SUBRECORD PRODUCT |
| 00039 | | ITEM CATEGORY |
| 00040 | | ITEM DESCRIPTION |
| 00041 | | ITEM PRICE |
| 00042 | | ITEM QTY-ON-HAND |
| 00043 | | SUBFILE STOCK PASSWORD |
| 00044 | | CONTAINS PRODUCT |

Figure E-1. Data Definition for Defined File INVFILE



| LINE NO# | SEQ# | SOURCE STATEMENT |
|----------|------|--|
| 00001 | | IDENTIFICATION DIVISION |
| 00002 | | PROGRAM-ID= UNIQUE-2 |
| 00003 | | DATA DIVISION |
| 00004 | | FILE SECTION |
| 00005 | | FD CUSFILE |
| 00006 | | 01 CUST-REC |
| 00007 | | 02 CUST-NAME PIC X(18) |
| 00008 | | 02 CUST-ID PIC X(7) |
| 00009 | | 02 ADDRESS PIC X(20) |
| 00010 | | 02 CITY PIC X(11) |
| 00011 | | 02 STATE PIC X(2) |
| 00012 | | 02 ZIP PIC X(5) |
| 00013 | | 02 PHONE PIC X(12) |
| 00014 | | FD INVCFIL |
| 00015 | | 01 INVOICE-REC |
| 00016 | | 02 CUST-ID PIC X(7) |
| 00017 | | 02 INV-NO PIC X(6) |
| 00018 | | 02 INV-DATE PIC X(8) |
| 00019 | | 02 TOTAL-SALES PIC 9(7)V99 USAGE COMP-3 |
| 00020 | | 02 TAX-AMT PIC 9(5)V99 USAGE COMP-3 |
| 00021 | | 02 INV-GROSS PIC 9(7)V99 USAGE COMP-3 |
| 00022 | | 01 ITEM-REC |
| 00023 | | 02 CUST-ID PIC X(7) |
| 00024 | | 02 INV-NO PIC X(6) |
| 00025 | | 02 MMBR PIC X(2) |
| 00026 | | 02 ITEM-CODE PIC X(6) |
| 00027 | | 02 DESCRIPTION PIC X(18) |
| 00028 | | 02 QUANTITY PIC 9(3) USAGE COMP-3 |
| 00029 | | 02 UNIT-PRICE PIC 9(6)V99 USAGE COMP-3 |
| 00030 | | 02 TOT-PRICE PIC 9(7)V99 USAGE COMP-3 |
| 00031 | | DEFINITION DIVISION |
| 00032 | | DEFINED FILE SALES PASSWORD |
| 00033 | | DEFINED RECORD CUSTOMER FROM CUST-REC |
| 00034 | | ALLOW ADD AND DELETE |
| 00035 | | IDENTIFIER CUST-NAME |
| 00036 | | ITEM CUST-ID MUST ADD |
| 00037 | | ITEM ADDRESS ALLOW CHANGE MUST ADD |
| 00038 | | ITEM CITY ALLOW CHANGE MUST ADD |
| 00039 | | ITEM ST FROM STATE ALLOW CHANGE MUST ADD |
| 00040 | | ITEM ZIP ALLOW CHANGE |
| 00041 | | ITEM PHONE ALLOW CHANGE |
| 00042 | | DEFINED RECORD INVOICE FROM INVOICE-REC |
| 00043 | | ALLOW ADD AND DELETE |
| 00044 | | PARENT IS CUSTOMER |
| 00045 | | POINTER IS CUST-ID |
| 00046 | | IDENTIFIER INV-NO |
| 00047 | | ITEM INV-DATE MUST ADD |
| 00048 | | ITEM TOTAL-SALES ALLOW CHANGE MUST ADD |
| 00049 | | ITEM TAX-AMT ALLOW CHANGE |
| 00050 | | ITEM INV-GROSS ALLOW CHANGE MUST ADD |
| 00051 | | DEFINED RECORD INV-ITEM FROM ITEM-REC |
| 00052 | | ALLOW ADD AND DELETE |
| 00053 | | PARENT IS INVOICE |
| 00054 | | IDENTIFIER MMBR |
| 00055 | | ITEM ITEM-CODE MUST ADD |
| 00056 | | ITEM DESCRIPTION MUST ADD |
| 00057 | | ITEM QUANTITY ALLOW CHANGE MUST ADD |
| 00058 | | ITEM UNIT-PRICE ALLOW CHANGE MUST ADD |
| 00059 | | ITEM TOT-PRICE ALLOW CHANGE MUST ADD |

Figure E-2. Data Definition for Defined File SALES

Index

| Term | Reference | Page | Term | Reference | Page |
|--|-----------|------|---|-----------|------|
| A | | | | | |
| Action programs | | | ADD complete message | 7.11 | 7—15 |
| accessing defined files | 1.1 | 1—2 | AFTER/FROM clause (LIST command) | | |
| record areas for receiving defined records | 4.1 | 4—2 | examples | 7.21 | 7—44 |
| ADD command | | | format | 7.21 | 7—43 |
| display format | 7.11 | 7—15 | function | 7.21 | 7—43 |
| function | 7.10 | 7—15 | listing specific items | 7.21 | 7—44 |
| hard-copy format | 7.12 | 7—20 | listing whole records | 7.21 | 7—44 |
| update state | 7.10 | 7—15 | with IF and FOR clauses | 7.21 | 7—44 |
| ADD command (display format) | | | ALLOW ADD AND DELETE clause (defined record definition) | | |
| canceling update | 7.11 | 7—20 | example | 3.18 | 3—33 |
| correcting errors | 7.11 | 7—17 | format | 3.18 | 3—33 |
| description | 7.11 | 7—15 | purpose | 3.18 | 3—33 |
| example | 7.11 | 7—16 | use with MUST ADD option | 3.26 | 3—44 |
| format | 7.11 | 7—15 | ALLOW ADD AND DELETE clause (subrecord definition) | | |
| MUST ADD items | 7.11 | 7—18 | example | 3.46 | 3—70 |
| OK command | 7.11 | 7—19 | format | 3.46 | 3—69 |
| omitting items | 7.11 | 7—18 | purpose | 3.46 | 3—69 |
| requesting multiple records | 7.11 | 7—15 | ALLOW CHANGE option (defined record item) | | |
| transmitting update screen | 7.11 | 7—16 | example | 3.27 | 3—45 |
| update format display | 7.11 | 7—15 | format | 3.27 | 3—45 |
| update validation and error display | 7.11 | 7—16 | how it's used | 3.27 | 3—45 |
| ADD command (hard-copy format) | | | purpose | 3.27 | 3—45 |
| canceling update | 7.12 | 7—23 | restrictions on use | 3.27 | 3—45 |
| correcting errors | 7.12 | 7—23 | ALLOW CHANGE option (subitem) | | |
| description | 7.12 | 7—20 | example | 3.51 | 3—74 |
| entering values by position | 7.12 | 7—21 | format | 3.51 | 3—74 |
| examples | 7.12 | 7—21 | purpose | 3.51 | 3—74 |
| format | 7.12 | 7—20 | ALLOW CHANGE option (supplement item) | | |
| item-name/positional combination | 7.12 | 7—22 | example | 3.40 | 3—65 |
| naming items and values | 7.12 | 7—21 | format | 3.40 | 3—65 |
| OK command | 7.12 | 7—23 | ROLE IN UPDATE options | 3.35 | 3—59 |
| output display | 7.12 | 7—21 | rules | 3.40 | 3—65 |
| SHOW command | 7.12 | 7—21 | | | |
| subformats | 7.12 | 7—21 | | | |
| update validation | 7.12 | 7—22 | | | |

| Term | Reference | Page | Term | Reference | Page |
|------------------------------------|------------|------|-------------------------------|-----------|------|
| CONTROLLED ROLE IN UPDATE | | | Defined file definition | | |
| example | 3.35 | 3-60 | contents | 3.6 | 3-8 |
| function | 3.35 | 3-60 | format | Fig. 3-4 | 3-9 |
| interrelated defined files example | 4.5 | 4-14 | nested structure | 3.6 | 3-8 |
| restrictions | 3.35 | 3-60 | required and optional entries | 3.6 | 3-8 |
| CONTROLLING ROLE IN UPDATE | | | Defined file name | | |
| example | 3.35 | 3-60 | coding rule | 3.7 | 3-10 |
| format | 3.35 | 3-59 | outside references | 3.7 | 3-11 |
| function | 3.35 | 3-60 | DEFINED FILE statement | | |
| restrictions | 3.35 | 3-60 | coding | 3.7 | 3-10 |
| | | | examples | 3.7 | 3-11 |
| | | | format | 3.7 | 3-10 |
| | | | function | 3.7 | 3-10 |
| | | | passwords | 3.7 | 3-10 |
| | | | Defined files | | |
| | | | accessing through UNIQUE | 1.3 | 1-3 |
| | | | characteristics | 1.1 | 1-1 |
| | | | creation | 1.2 | 1-2 |
| | | | hierarchical structure | 2.4 | 2-3 |
| | | | multiple, example | 4.5 | 4-14 |
| | | | naming | 3.7 | 3-10 |
| | | | simple | 2.4 | 2-3 |
| | | | sources | 2.1 | 2-1 |
| | | | structure | 2.1 | 2-2 |
| | | | variations | 3.53 | 3-76 |
| | | | DEFINED RECORD clause | | |
| | | | coding rule | 3.10 | 3-15 |
| | | | examples | 3.10 | 3-15 |
| | | | format | 3.10 | 3-15 |
| | | | function | 3.10 | 3-15 |
| | | | Defined record definition | | |
| | | | contents | 3.8 | 3-12 |
| | | | DEFINED RECORD clause | 3.10 | 3-15 |
| | | | format | Fig. 3-5 | 3-12 |
| | | | function | 3.8 | 3-12 |
| | | | item definition | 3.19 | 3-34 |
| | | | subrecord definition | 3.43 | 3-67 |
| | | | supplement definition | 3.30 | 3-48 |
| | | | Defined record items | | |
| | | | allowing changes | 3.27 | 3-45 |
| | | | defining | 3.23 | 3-41 |
| | | | function | 3.19 | 3-34 |
| | | | naming | 3.24 | 3-42 |
| | | | preventing display | 3.25 | 3-43 |
| | | | previously defined items | 3.29 | 3-47 |
| | | | specifying a required item | 3.26 | 3-44 |
| | | | value ranges | 3.28 | 3-46 |
| | | | Defined record management | | |
| | | | characteristics | 1.4 | 1-3 |
| | | | constructing defined records | 1.4 | 1-4 |
| | | | validating updates | 1.4 | 1-4 |
| D | | | | | |
| Data definition processor | | | | | |
| characteristics | 5.1 | 5-1 | | | |
| diagnostics issued | Appendix C | | | | |
| error processing | 5.5 | 5-8 | | | |
| function | 1.2 | 1-2 | | | |
| job control stream | 5.1 | 5-2 | | | |
| options | 5.2 | 5-2 | | | |
| output listings | 5.4 | 5-7 | | | |
| sample job control streams | 5.3 | 5-4 | | | |
| Data definition records | | | | | |
| contents | 2.3 | 2-3 | | | |
| creation and storage | 2.3 | 2-3 | | | |
| Data definitions | | | | | |
| conventions and formats | Appendix A | | | | |
| data division | 3.4 | 3-4 | | | |
| definition division | 3.5 | 3-7 | | | |
| elements | 1.2 | 1-2 | | | |
| identification division | 3.3 | 3-4 | | | |
| overall format | Fig. 3-1 | 3-3 | | | |
| reserved words | Appendix B | | | | |
| sources | 3.1 | 3-1 | | | |
| structure | 3.2 | 3-3 | | | |
| using Katakana characters | 3.1 | 3-2 | | | |
| Data division | | | | | |
| contents | 3.4 | 3-4 | | | |
| example | 3.4 | 3-5 | | | |
| FD statements | 3.4 | 3-5 | | | |
| record descriptions | Fig. 3-2 | 3-6 | | | |
| required entries | 3.4 | 3-5 | | | |

| Term | Reference | Page | Term | Reference | Page |
|--|-----------|------|-------------------------------------|-----------|------|
| F | | | H | | |
| FD statements | 3.4 | 3—5 | Hard-copy format (UNIQUE commands) | | |
| FILL KEY clause (defined record definition) | | | ADD command | 7.12 | 7—20 |
| example | 3.17 | 3—32 | CANCEL command | 7.9 | 7—13 |
| format | 3.17 | 3—31 | CHANGE command | 7.15 | 7—27 |
| how they work | 3.17 | 3—31 | OK command | 7.8 | 7—13 |
| purpose | 3.17 | 3—31 | HIDDEN option (defined record item) | | |
| when to use | 3.17 | 3—31 | example | 3.25 | 3—44 |
| FILL KEY clause (supplement) | | | format | 3.25 | 3—43 |
| examples | 3.34 | 3—58 | how it's used | 3.25 | 3—43 |
| format | 3.34 | 3—58 | purpose | 3.25 | 3—43 |
| how they work | 3.34 | 3—58 | HIDDEN option (supplement) | | |
| purpose | 3.34 | 3—56 | example | 3.38 | 3—64 |
| summary of uses | Table 3—2 | 3—57 | format | 3.38 | 3—63 |
| when to use | 3.34 | 3—57 | rules | 3.38 | 3—63 |
| FOLLOWS clause | | | Hierarchical defined files | | |
| example | 3.16 | 3—30 | characteristics | 2.4 | 2—3 |
| format | 3.16 | 3—29 | examples | 4.2 | 4—4 |
| function | 3.16 | 3—29 | fraternal records | 2.4 | 2—4 |
| restrictions on use | 3.16 | 3—29 | identifiers | 3.19 | 3—34 |
| FOR clause (LIST command) | | | parent-child records | 2.4 | 2—4 |
| AFTER/FROM clauses | 7.22 | 7—45 | PARENT clause | 3.13 | 3—22 |
| examples | 7.20 | 7—41 | POINTER clause | 3.15 | 3—27 |
| format | 7.20 | 7—40 | PREFIX clause | 3.14 | 3—25 |
| function | 7.20 | 7—40 | vs simple defined files | 3.19 | 3—36 |
| IF clause | 7.20 | 7—40 | | | |
| listing child records | 7.20 | 7—41 | | | |
| listing parent records with same child record | Fig. 3—6 | 3—16 | | | |
| listing records with same major identifier | Fig. 3—20 | 3—39 | | | |
| listing specific items | 7.20 | 7—42 | | | |
| listing two record types | 7.20 | 7—41 | | | |
| Fraternal records | | | | | |
| example | 2.4 | 2—4 | | | |
| PREFIX clause | 3.14 | 3—25 | | | |
| VALUE clause | 3.22 | 3—41 | | | |
| FROM clause (defined record definition) | | | | | |
| function | 3.11 | 3—16 | | | |
| specifying one record as source | 3.11 | 3—16 | | | |
| specifying repeating group as source | 3.11 | 3—20 | | | |
| specifying sequence of records as source | 3.11 | 3—17 | | | |
| FROM clause (supplement definition) | | | | | |
| function | 3.32 | 3—50 | | | |
| specifying one record as source | 3.32 | 3—50 | | | |
| specifying repeating group as source | 3.32 | 3—51 | | | |

| Term | Reference | Page | Term | Reference | Page |
|---------------------------------------|-----------|------|----------------------------------|-----------|------|
| Identification division | | | ITEM clause (supplement) | | |
| example | 3.3 | 3-4 | example | 3.37 | 3-63 |
| optional entries | 3.3 | 3-4 | format | 3.37 | 3-63 |
| required entries | 3.3 | 3-4 | rules | 3.37 | 3-63 |
| IDENTIFIER clause (defined record) | | | ITEM definition (defined record) | | |
| example | 3.21 | 3-38 | format | Fig. 3-14 | 3-34 |
| format | 3.21 | 3-38 | function | 3.19 | 3-34 |
| function | 3.21 | 3-38 | IDENTIFIER statements | 3.20 | 3-37 |
| multiple identifiers | 3.21 | 3-38 | ITEM statements | 3.23 | 3-41 |
| IDENTIFIER clause (subrecord) | | | Item definition (supplement) | | |
| example | 3.48 | 3-72 | ALLOW CHANGE option | 3.40 | 3-65 |
| format | 3.48 | 3-71 | ALSO clause | 3.42 | 3-66 |
| purpose | 3.48 | 3-71 | contents | 3.36 | 3-62 |
| Identifier items (defined record) | | | format | 3.36 | 3-62 |
| defining | 3.20 | 3-37 | HIDDEN option | 3.38 | 3-63 |
| function | 3.19 | 3-34 | ITEM clause | 3.37 | 3-63 |
| hierarchical defined files | 3.19 | 3-36 | MUST ADD option | 3.39 | 3-64 |
| multiple identifiers | 3.21 | 3-38 | VALUE clause | 3.41 | 3-65 |
| naming | 3.21 | 3-38 | ITEM statement (defined record) | | |
| simple defined files | 3.19 | 3-35 | ALLOW CHANGE option | 3.27 | 3-45 |
| value range | 3.22 | 3-40 | ALSO clause | 3.29 | 3-47 |
| IDENTIFIER statement (defined record) | | | format | 3.23 | 3-41 |
| format | 3.20 | 3-37 | HIDDEN option | 3.25 | 3-43 |
| IDENTIFIER clause | 3.21 | 3-38 | ITEM clause | 3.24 | 3-42 |
| VALUE clause | 3.22 | 3-40 | MUST ADD option | 3.26 | 3-44 |
| IF clause (LIST command) | | | VALUE clause | 3.28 | 3-46 |
| AFTER/FROM clause | 7.22 | 7-45 | Items (defined record) | | |
| comparing item with literal | 7.19 | 7-38 | allowing changes | 3.27 | 3-45 |
| comparing two record items | 7.19 | 7-38 | defining | 3.23 | 3-41 |
| comparison operators | 7.19 | 7-37 | function | 3.19 | 3-34 |
| complex conditional expression | 7.19 | 7-39 | naming | 3.24 | 3-42 |
| examples | 7.19 | 7-38 | preventing display | 3.25 | 3-43 |
| FOR clause | 7.20 | 7-40 | previously defined items | 3.29 | 3-47 |
| format | 7.19 | 7-37 | specifying a required item | 3.26 | 3-44 |
| forming value range | 7.19 | 7-40 | value range | 3.28 | 3-46 |
| function | 7.19 | 7-37 | Items (supplement) | | |
| requiring two conditions | 7.19 | 7-40 | allowing changes | 3.40 | 3-65 |
| simple conditional expression | 7.19 | 7-37 | naming | 3.37 | 3-63 |
| ITEM clause (defined record) | | | preventing display | 3.38 | 3-63 |
| example | 3.24 | 3-42 | previously defined items | 3.42 | 3-66 |
| format | 3.24 | 3-42 | sources | 3.30 | 3-48 |
| function | 3.24 | 3-42 | specifying a required item | 3.39 | 3-64 |
| ITEM clause (subrecord) | | | value range | 3.41 | 3-65 |
| example | 3.49 | 3-73 | | | |
| format | 3.49 | 3-72 | | | |
| function | 3.49 | 3-72 | | | |

| Term | Reference | Page | Term | Reference | Page |
|---|-----------|------|---------------------------------------|-----------|------|
| J | | | M | | |
| Job control stream (data definition processor) | | | MORE command | | |
| contents | 5.1 | 5-1 | embedded | 7.23 | 7-48 |
| options | 5.2 | 5-2 | examples | 7.24 | 7-49 |
| sample streams | 5.3 | 5-4 | format | 7.23 | 7-48 |
| | | | function | 7.23 | 7-48 |
| | | | requesting next screen later | 7.23 | 7-48 |
| | | | with LIST or DETAIL | 7.23 | 7-48 |
| | | | without LIST or DETAIL | 7.23 | 7-48 |
| | | | Multiple defined files | | |
| | | | example | 4.5 | 4-14 |
| | | | storing in NAMEREC | 5.1 | 5-1 |
| | | | MUST ADD option (defined record item) | | |
| | | | example | 3.26 | 3-44 |
| | | | format | 3.26 | 3-44 |
| | | | how it's used | 3.26 | 3-44 |
| | | | purpose | 3.26 | 3-44 |
| | | | UNIQUE ADD command | 7.11 | 7-18 |
| | | | MUST ADD option (subitem) | | |
| | | | example | 3.50 | 3-73 |
| | | | format | 3.50 | 3-73 |
| | | | purpose | 3.50 | 3-73 |
| | | | restrictions | 3.50 | 3-73 |
| | | | UNIQUE ADD command | 7.11 | 7-18 |
| | | | MUST ADD option (supplement item) | | |
| | | | example | 3.39 | 3-64 |
| | | | format | 3.39 | 3-64 |
| | | | ROLE IN UPDATE options | 3.35 | 3-59 |
| | | | rules | 3.39 | 3-64 |
| | | | UNIQUE ADD command | 7.11 | 7-18 |
| K | | | | | |
| Katakana characters | | | | | |
| specifying in PARAM option | 5.2 | 5-3 | | | |
| using in data definition | 3.1 | 3-2 | | | |
| Keywords, list command | Table D-3 | D-3 | | | |
| L | | | | | |
| Lexicon (UNIQUE) | | | | | |
| list command keywords | Table D-3 | D-3 | | | |
| list command logical operators | Table D-4 | D-3 | | | |
| list command statistical functions | Table D-5 | D-3 | | | |
| punctuation in UNIQUE commands | Table D-2 | D-2 | | | |
| UNIQUE commands | Table D-1 | D-2 | | | |
| words in UNIQUE status and error messages | Table D-6 | D-4 | | | |
| LIST command | | | | | |
| AFTER/FROM clause | 7.21 | 7-43 | | | |
| display-content-specification | 7.18 | 7-31 | | | |
| FOR clause | 7.20 | 7-40 | | | |
| format | 7.16 | 7-30 | | | |
| function | 7.16 | 7-30 | | | |
| IF clause | 7.19 | 7-37 | | | |
| parameters | 7.16 | 7-30 | | | |
| statistical-function | 7.22 | 7-45 | | | |
| unqualified LIST | 7.17 | 7-30 | | | |
| Logical operators, list command | Table D-4 | D-3 | | | |

| Term | Reference | Page | Term | Reference | Page |
|------------------------------|-----------|------|--|-----------|------|
| N | | | O | | |
| Named record file (NAMEREC) | | | OF clause | | |
| initializing | 5.1 | 5—2 | example | 3.45 | 3—69 |
| purpose | 5.1 | 5—1 | format | 3.45 | 3—69 |
| NAMEREC utility | | | purpose | 3.45 | 3—69 |
| passwords for defined files | 3.7 | 3—10 | OK command | | |
| passwords for subfiles | 3.54 | 3—77 | completion message | 7.8 | 7—13 |
| NEUTRAL ROLL IN UPDATE | | | format | 7.8 | 7—13 |
| example | 3.35 | 3—62 | function | 7.8 | 7—13 |
| function | 3.35 | 3—61 | hard-copy terminal requirement | 7.8 | 7—13 |
| restrictions | Table 3—3 | 3—60 | One record as source of defined record | | |
| NEXT command | | | example | 3.11 | 3—17 |
| CANCEL command | 7.9 | 7—13 | format | 3.11 | 3—16 |
| DISPLAY COMPLETE message | 7.6 | 7—10 | purpose | 3.11 | 3—17 |
| displaying next record later | 7.6 | 7—10 | One record as source of supplement | | |
| embedded NEXT | 7.6 | 7—9 | example | 3.32 | 3—51 |
| example | 7.6 | 7—10 | format | 3.32 | 3—50 |
| format | 7.6 | 7—9 | purpose | 3.32 | 3—50 |
| function | 7.6 | 7—9 | OPEN command | | |
| OK command | 7.8 | 7—13 | defined file passwords | 3.7 | 3—10 |
| | | | example | 7.3 | 7—5 |
| | | | format | 7.3 | 7—5 |
| | | | function | 7.3 | 7—5 |
| | | | subfile passwords | 3.54 | 3—77 |
| | | | Output listing | | |
| | | | (data definition processor) | | |
| | | | contents | 5.4 | 5—7 |
| | | | diagnostic messages | 5.5 | 5—8 |
| | | | item status bytes | 5.4 | 5—7 |

| Term | Reference | Page | Term | Reference | Page |
|--|-----------|------|------------------------------|-----------|------|
| S | | | | | |
| Simple defined files | | | Subfiles | | |
| example | 4.1 | 4—2 | contents | 3.53 | 3—76 |
| identifiers | 3.19 | 3—35 | example | 4.4 | 4—12 |
| vs hierarchical defined files | 3.19 | 3—36 | function | 3.53 | 3—76 |
| SHOW command | | | identifying included records | 3.55 | 3—78 |
| format | 7.25 | 7—52 | naming | 3.54 | 3—76 |
| function | 7.25 | 7—52 | UNIQUE example | 7.18 | 7—31 |
| record format display | 7.25 | 7—52 | SUBITEM definition | | |
| symbols used in examples | Table 7—1 | 7—52 | ALLOW CHANGE option | 3.51 | 3—74 |
| update formats | 7.25 | 7—52 | format | Fig. 3—27 | 3—67 |
| Source file, naming in PARAM statement | 5.2 | 5—2 | function | 3.47 | 3—70 |
| Sources of defined records | | | IDENTIFIER clause | 3.48 | 3—71 |
| describing | 3.4 | 3—4 | ITEM clause | 3.49 | 3—72 |
| one record as source | 3.11 | 3—16 | MUST ADD option | 3.50 | 3—73 |
| repeating group as source | 3.11 | 3—20 | VALUE clause | 3.52 | 3—75 |
| sequence of records as source | 3.11 | 3—17 | Subitems | | |
| Sources of supplements | | | allowing changes | 3.51 | 3—74 |
| one record as source | 3.32 | 3—50 | independence | 3.43 | 3—68 |
| repeating group as source | 3.32 | 3—51 | naming items | 3.49 | 3—72 |
| Statistical-function (LIST command) | | | renaming identifiers | 3.48 | 3—71 |
| example | 7.22 | 7—46 | specifying a required item | 3.50 | 3—73 |
| format | 7.22 | 7—45 | value range | 3.52 | 3—75 |
| function | 7.22 | 7—45 | SUBRECORD clause | | |
| hierarchical file | 7.22 | 7—46 | example | 3.44 | 3—68 |
| item-names | 7.22 | 7—45 | format | 3.44 | 3—68 |
| statistical functions | 7.22 | 7—45 | function | 3.44 | 3—68 |
| UNIQUE lexicon | Table D—5 | D—3 | Subrecord definition | | |
| Subfile definition | | | ALLOW ADD AND DELETE clause | 3.46 | 3—69 |
| CONTAINS clause | 3.55 | 3—78 | format | Fig. 3—27 | 3—67 |
| contents | 3.53 | 3—76 | OF clause | 3.45 | 3—69 |
| format | Fig. 3—28 | 3—71 | purpose | 3.43 | 3—67 |
| purpose | 3.53 | 3—76 | subitem definition | 3.47 | 3—70 |
| SUBFILE statement | 3.54 | 3—76 | SUBRECORD clause | 3.44 | 3—68 |
| Subfile name | | | Subrecords | | |
| coding rules | 3.54 | 3—76 | allowing subrecord additions | | |
| outside references | 3.54 | 3—77 | and deletions | 3.46 | 3—69 |
| SUBFILE statement | | | contents | 3.43 | 3—67 |
| example | 3.54 | 3—78 | defining subrecord items | 3.47 | 3—70 |
| format | 3.54 | 3—76 | example | 4.4 | 4—12 |
| function | 3.54 | 3—76 | identifiers | 3.43 | 3—67 |
| passwords | 3.54 | 3—77 | independence | 3.43 | 3—68 |
| | | | naming | 3.44 | 3—68 |
| | | | previously defined items | 3.45 | 3—69 |
| | | | subfiles | 3.43 | 3—67 |
| | | | UNIQUE example | 7.18 | 7—31 |
| | | | SUPPLEMENT clause | | |
| | | | example | 3.31 | 3—49 |
| | | | format | 3.31 | 3—49 |
| | | | function | 3.31 | 3—49 |

| Term | Reference | Page | Term | Reference | Page |
|---|-----------|------|--------------------------------|-----------|------|
| V | | | | | |
| VALUE clause (defined record identifier) | | | VALUE clause (subitem) | | |
| effect of omitting | 3.22 | 3-40 | example | 3.52 | 3-75 |
| example | 3.22 | 3-41 | format | 3.52 | 3-75 |
| file segmentation | 3.22 | 3-41 | purpose | 3.52 | 3-75 |
| format | 3.22 | 3-40 | VALUE clause (supplement item) | | |
| fraternal records | 3.22 | 3-41 | example | 3.41 | 3-66 |
| how it's used | 3.22 | 3-40 | format | 3.41 | 3-65 |
| purpose | 3.22 | 3-40 | rules | 3.41 | 3-65 |
| VALUE clause (defined record item) | | | | | |
| effect of omitting | 3.28 | 3-46 | | | |
| example | 3.28 | 3-46 | | | |
| format | 3.28 | 3-46 | | | |
| purpose | 3.28 | 3-46 | | | |

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

(Document Title)

(Document No.)

(Revision No.)

(Update No.)

Comments:

Cut along line.

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424



CUT

FOLD