
Open M with SQL Data Dictionary Guide

Version: Open M with SQL F.10

Revision Date: April 18, 1997

Legacy Archive Acrobat File Generated on: January 22, 2004

Part Number

IS-SQL-8-F.10A-CP-B

Print History

Creation Date:	June 1991
Revision Date:	August 1992
	June 1994
	March 1997

Open M with SQL Data Dictionary Guide

Copyright © InterSystems Corporation

1997

All rights reserved

NOTICE

PROPRIETARY — CONFIDENTIAL

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

M/SQL®, *M/PACT®*, and *M/NET®* are registered trademarks, and *InterSystems™*, *Open M™*, *Open M/SQL™*, *ISM™*, *DTM™*, *DT-MAX™*, *DT Windows™*, *DSM™*, and *DASL™* are trademarks of InterSystems Corporation.

DSM DDP™, *VAX™*, *VMS™*, *Open VMS™*, and *DEC™* are trademarks of Digital Equipment Corp.

Microsoft®, *MS-DOS®*, *Microsoft Access®*, and, *Excel®* are registered trademarks and *Windows™*, *Windows NT*, *Visual Basic™*, and *Visual C++™* are trademarks of Microsoft Corporation.

ORACLE® is a registered trademark of Oracle Corporation

For Support questions about any InterSystems products, contact the InterSystems Worldwide Support Center:

Phone: US: +1 617 621-0700

Europe: +44 (0) 1753 830-077

Fax: US: +1 617 374-9391

Europe: +44 (0) 1753 861-311

Internet — support@intersys.com

FTP Site — ftp.intersys.com

World Wide Web — www.intersys.com

BBS: General Use: +1 (617) 225-0475
Developers: +1 (617) 494-0867

Europe: +44 (0) 1753-853-534

Preface

The *Open M with SQL Data Dictionary Guide* describes the Open M with SQL Data Dictionary. You use the Data Dictionary to define tables to represent the logical structure of a relational database, or map the physical structure of an existing database to relational tables. These tables act as the foundation on which Open M with SQL relational database applications are built.

Open M with SQL combines two ANSI-standard languages — M, the only procedural programming language designed specifically for database applications, and SQL (Structured Query Language), the most widely used relational query language.

M is the name for an ANSI-standard database-oriented procedural language formerly known as MUMPS. Open M is the InterSystems implementation of that language. As with the other components of the Open M with SQL relational database application development system, you can use the Data Dictionary with Open M, or other vendors' versions of M.

Open M with SQL is an integrated environment for developing and running database applications. It includes an advanced relational database management system, an application and report generator, and a procedural programming language.

Open M with SQL provides two program development environments:

- The M environment, in which you can execute system management commands, and create, modify, and execute M routines.
- The Open M Developer relational environment, in which you define the database in relational terms and use the application generator tools to create, modify, and execute advanced relational database applications.

This document is specifically concerned with the relational environment.

Open M with SQL's relational environment is a complete RDBMS (Relational Database Management System)—it includes a development environment for creating advanced relational database applications, a management system for maintaining them, and a run-time environment for executing them.

The other guides which complete the description of the RDBMS application development system are *Open M/SQL: A Gentle Introduction*, the *Open M/SQL Developer Guide*, the *Open M/SQL Form Generator Guide*, and the *Open M/SQL M/PACT Guide*.

Refer to the *M Language Reference Guide* for a complete description of Inter-Systems' Open M with SQL M and DTM. See the *Open M/SQL M Programming Guide* to learn how to develop applications using Open M. Refer to the Open M with SQL System Guide for your platform to learn about running Open M on your system. See the *DataTree Programmer's Guide* to learn how to develop applications using DTM.

Refer to the *Open M/SQL Server Programming Guide* and the *Open M/SQL Server User Guide* to learn about the Open M with SQL Relational Server product and how you can use it to access your M database from third party applications.

Audience

This guide is designed for use by developers of relational database applications. It assumes that you have M and SQL programming experience and understand your own application's database requirements.

This document describes how to configure, manage, and maintain an Open M with SQL relational database environment, including both the application development environment and the application run-time environment. Its goal is to help you achieve a smoothly operating Open M with SQL system.

For sites that use the Open M with SQL Relational Server to make Open M with SQL data available to external applications, this document also describes how to configure, manage, and maintain the Relational Server, with its goal to help you achieve a smoothly operating Server environment.

Organization of This Guide

This guide is divided into four parts, plus appendices, a glossary, and an index.

Part I, “Introduction,” contains the following four chapters:

- n Chapter 1, *Introduction to the Data Dictionary*
- n Chapter 2, *The Open M with SQL Relational Database*
- n Chapter 3, *Using the Data Dictionary Interface*
- n Chapter 4, *Designing a Sample Application*

Part II, “Basic Operations,” contains the following eleven chapters:

- n Chapter 5, *Defining a Base Table*
- n Chapter 6, *Defining Base Table Fields*
- n Chapter 7, *Implicit Joins*
- n Chapter 8, *Base Table Lookups*
- n Chapter 9, *Index Maps*
- n Chapter 10, *Data Conversion and Validation*
- n Chapter 11, *Base Table Triggers*
- n Chapter 12, *Base Table Help and Error Messages*
- n Chapter 13, *Field Help and Error Messages*
- n Chapter 14, *Views*
- n Chapter 15, *Data Dictionary Reports*

Part III, “Physical Structure” contains the following three chapters:

- n Chapter 16, *Default Physical Structure*
- n Chapter 17, *Relational Definition of an M Database*
- n Chapter 18, *Creating a Customized Map Definition*

Part IV, “Special Topics” contains the following two chapters:

- n Chapter 19, *The FileMan Interface*
- n Chapter 20, *Importing Data Definitions*

The three Appendices to this guide are as follows:

- ⁂ Appendix A, *Data Dictionary Specifications*, which lists the maximum values for base table and field parameters.
- ⁂ Appendix B, *Keyboard Actions*, which shows the keyboard mapping of form actions for all supported keyboards.
- ⁂ Appendix C, *SQL Reserved Words*, which lists reserved words which should *not* be used as names base tables, views, or fields.
- ⁂ Appendix D, *Base Table Design Charts*, which provides charts for designing the base tables in a relational application.

A glossary defines the terminology used in this guide.

Other References

Depending on the configuration of your system, you may also wish to refer to the following guides:

- n The *Open M/SQL Developer's Guide*, which describes the Open M with SQL relational database program development capabilities, including development of macro source routines and use of routine editing utilities, as well as Open M with SQL's implementation of SQL and the various methods for using SQL to query an Open M with SQL relational database.
- n The *Open M/SQL User Interface Programming Guide*, which describes how to use the application generation tools within Open M Developer to create a user interface for your relational database. These application generation tools include the **Form Generator** — used to create ready-to-run data screens through which end users can add, retrieve, edit, and delete database information; the **Menu Generator** — used to create pop-up menus and menu bars which unite the various components of an application in a logical and visually sophisticated manner and structure an application by defining how it is organized and how it is presented to users; and the **Application Help Facility** — used to create and deploy a context-sensitive on-line help system for your Open M with SQL relational database application.
- n The *Open M/SQL Relational Client User Guide*, which describes how to access an Open M with SQL relational database (retrieve and modify data) from various third-party application development tools.
- n The *Open M/SQL Server Programming Guide*, which describes how to access an Open M with SQL relational database from applications created using the C or C++ programming language.
- n The *M/PACT Guide*, which describes how to create and run sophisticated end-user data reporting applications that interact with the Data Dictionary.
- n An tutorial entitled *Open M/SQL: A Gentle Introduction*, which uses a project management demonstration application to show how to develop and use an Open M with SQL application.
- n The *Open M/SQL M Programming Guide*, which describes ISM (InterSystems' original implementation of the M programming language), including global database concepts, the development of intermediate code routines, and the use of ISM utilities.
- n The appropriate system guide for your Open M system platform, which describes the system-specific elements of using Open M on your particular computer and operating system.

Typographic Conventions Used in This Guide

This guide observes the following typographic conventions:

Convention Description	Example
Body text appears in Times Roman type.	The Database Administrator sets all system configuration parameters via the System Configuration window.
Computer generated information appears in Courier type.	Linking/Syntax Checking...DONE
User input appears in Courier italic type.	<i>Yes</i>
Keystrokes appear in upper-case and enclosed within angle brackets.	<RETURN>
Simultaneous keystrokes appear hyphenated, in upper-case, and enclosed within angle brackets.	<CTRL-Z>
Procedure titles appear in bold Helvetica type. Procedures and Examples are designated by an underlined sidehead prefix.	<u>Procedure</u>To access the System Configuration window:
Note, Caution, and Warning statements appear in Helvetica type with an appropriate sidehead.	Note Turning DBMS security OFF is a useful way to guarantee access to current applications by users who will need them while you are in the process of implementing a privilege system.
Titles of other guides and other chapters within this guide appear in Times Italic type.	<i>Open M/SQL User Interface Programming Guide</i>

Table of Contents

Preface

Audience	v
Organization of This Guide	vi
Other References	viii
Typographic Conventions Used in This Guide	ix

Part I—Introduction

1 Introduction to the Data Dictionary

The Open M with SQL Application Development Environment.	1-2
Defining a Relational Database in the Data Dictionary.	1-2
Open M with SQL Combines Two ANSI-Standard Languages: SQL and M 1-3	
M Language	1-3
Open M Developer.	1-4
Open M with SQL Server Opens Your Relational Database.	1-5
How is Open M with SQL “Open”	1-5
Developing Relational Applications	1-6
Defining a Relational Database in the Data Dictionary	1-7
Mapping the Functional Specifications of Your Database	1-7
Defining the Data Structure in the Data Dictionary	1-8
Defining Existing M Databases Relationally.	1-8
Accessing Data	1-9
Defining Base Tables in the Data Dictionary	1-9
Defining Database Fields in Base Tables	1-11
Defining Views in the Data Dictionary	1-12
Mapping a Relational Database to M Globals	1-13
Selecting Default or Customized Physical Structure.	1-13

When to Use Default Physical Structure.	1-14
When to Use Customized Physical Structure	1-14
The Data Dictionary Provides Many Benefits.	1-15
Developer Productivity	1-15
Existing Applications	1-16
Referential Integrity	1-16
2 The Open M with SQL Relational Database	
Relational Database Is a Collection of Tables.	2-2
Open M with SQL Supports Two Types of Tables	2-3
Base Tables Contain Data Stored on Disk	2-4
Virtual Tables Exist in Temporary Memory.	2-4
A View Produces a Virtual Table	2-5
Open M with SQL Tables Follow First Normal Form	2-6
Open M with SQL Implements Schemas as M Directories.	2-8
Open M with SQL Database Structure for Non-ISM Implementations of M	2-8
Accessing Data in an Open M with SQL Relational Database	2-9
Using SQL to Query the Database	2-9
Cartesian Product	2-10
Project	2-12
Restrict	2-13
Joins	2-14
InterSystems' Extensions to the Relational Model	2-16
One-Way Outer Joins	2-16
Implicit Joins.	2-18
Multi-Line Fields	2-23
3 Using the Data Dictionary Interface	
Accessing the Data Dictionary	3-2
Understanding the Data Dictionary Interface	3-5
Single-Row Forms	3-5
Multi-Row Forms	3-5
Accessing the Base Table Definition Window	3-5
Elements of a Window.	3-7
Navigating in Forms	3-9
Keyboard Actions	3-9
Using On-Line Help	3-11
Saving a Base Table Definition and Exiting the Form	3-13
Save Menu.	3-14

Save on PROCEED Menu	3-14
--------------------------------	------

4 Designing a Sample Application

Designing a Relational Database	4-2
Mapping the Functional Specifications	4-2
Defining the Data Structure in the Data Dictionary	4-3
Overview of Sample Application	4-4
Sample Application Uses Default Physical Structure	4-4
Base Tables in Documentation Tracking System	4-5
Relationships Between Base Tables	4-6
Base Table Fields	4-7
Reports in Documentation Tracking System	4-15

Part II—Basic Operations

5 Defining a Base Table

Base Table Definition Overview	5-2
Base Table Definition Checklist	5-2
Defining a Base Table	5-4
Horizontal Options Menu	5-8
Defining a Child Table	5-11
Creating a Child Table from the Parent Table	5-11
Defining a Designative Reference to Child Table	5-14
Generating Default Physical Structure	5-15
Edit the Default Global Name	5-15
Open M with SQL Generates Structure Automatically	5-15
Physical Structure Updates Automatically	5-16
Row ID Field Created	5-16
Compiling a Base Table	5-17
Compilation Produces Routines	5-17
Base Table Routine Names	5-17
Routine Size	5-18
Number of Routines	5-19
Compiling Related Objects	5-20
Editing a Base Table Definition	5-21
Do Not Change Global Name Once Data Entered	5-21
You Cannot Change Status of a Child Table	5-21
Pre-Delete Trigger Updated if Parent Reference Edited	5-21
Deleting a Base Table Definition	5-23

Restrictions on Deleting a Base Table Definition	5-23
--	------

6 Defining Base Table Fields

Field Definition Overview	6-2
Field Definition Checklist	6-3
Field Definition Checklist	6-3
Steps 1 & 2: Specify a Field Name and Description	6-4
Step 3: Specify the Data Type	6-9
Exercise Care When Mapping to an Existing M Database	6-10
Defining a Date Field	6-11
Defining a Designative Reference Field	6-14
Defining a Multiple Choice Field	6-16
Defining a Name Field	6-18
Defining a Time Field	6-25
Defining a Yes/No Field	6-27
Collation Sequence Affects Name and Text Data Types	6-28
Numbers Come First	6-28
EXACT	6-28
ALPHAUP	6-29
Minus	6-29
Plus	6-30
Space	6-30
UPPER	6-30
Effects of Collation Sequence	6-31
Step 4: Specify Maximum Length of Data	6-33
Step 5: Specify If Values Must be Unique	6-34
Index Maps for Unique Fields	6-34
Step 6: Specify the Number of Distinct Values	6-35
Enter a Number	6-35
Enter NUMROWS	6-35
Use Rows in Designated Table for Designative Reference Fields	6-36
Step 7: Specify If the Field is Multi-Line	6-37
Displaying Multi-Line Fields on Forms and Reports	6-37
Multi-Line Fields May be Computed Fields	6-37
Two Storage Modes	6-37
Referencing Multi-Line Fields	6-39
Step 8: Define Computed Field Calculations	6-42
Benefits and Limitations of Computed Fields	6-42
Two Types of Computed Fields	6-42
Step 9: Define User Update Features	6-46
Defining Field Protection	6-48

Defining a Required Field	6-49
Defining Default Field Values	6-50
Examples of Default Values	6-51
Step 10: Define Data Conversion and Validation Code	6-52
Step 11: Define Error and Help Messages	6-52
Step 12: Define Additional Options	6-52
Define Column Titles for Reports	6-54
Step 13: Save the Field Definition	6-56
Copying a Field Definition	6-57
Editing and Deleting a Field Definition	6-59
Deleting a Field Definition	6-59

7 Implicit Joins

Use Designative Reference to Join Independent Tables	7-2
Defining a Designative Reference Field	7-2
Tables Joined by Designative Reference are Independent	7-2
Many-to-One Relationship	7-2
Relational Foreign Key Implemented as a Pointer to Row ID	7-2
Designative Reference is a One Way Outer Join	7-3
Multiple Designative Reference Fields	7-4
Designative Reference to a Child Table	7-4
Designative References from Multiple Tables	7-4
Use Characteristic Relationship to Join Dependent Tables	7-5
Automatically Created Trigger Deletes Child Rows	7-5
One-to-many Parent to Child Relationship	7-5
Characteristic Relationship is an Inner Join	7-5
Multi-Generation Parent-Child Relationships	7-5
Benefits of Implicit Joins	7-7
Extended Arrow Syntax	7-7
Referential Integrity Constraints	7-8
Efficient Data Access	7-8

8 Base Table Lookups

What is a Lookup?	8-1
Where to Define Lookups	8-3
Lookups and Index Maps	8-3
Lookup Specifications	8-3
Lookup Fields	8-3
Lookup Display Fields	8-4
Types of Lookup Queries	8-5

Unqualified Lookup Query	8-5
Single Field Lookup Query	8-5
Compound Lookup Query	8-6
Enabling Lookup Queries	8-6
Precedence Given to Most Restrictive Queries.	8-6
Defining Base Table Lookups	8-7
Designative Display Fields in Lookup Specifications	8-10
Defining Lookups for a Child Table	8-13
Two Types of Lookups	8-13
Matching	8-14
Field Matching Functions	8-15
Field Conversion Code	8-17
Summary of Case Transformation and Field Conversion Code	8-18
Customizing a Lookup Query	8-18
You Can Toggle Between Default and Override Versions.	8-19
Suppress Lookup Box	8-19
Lookup Queries Use InterSystems' SQL Extensions	8-21
 9 Index Maps	
Index Maps Speed Row Selection	9-2
Advantages of Indexing	9-2
Disadvantages of Indexing	9-2
Structure of Index Maps	9-3
Index Field and Row ID Values are Subscripts	9-3
Index Maps are Small	9-3
Rows with Null Value In Index Field Not Included	9-3
Make Indexed Fields Required to Avoid Null Values	9-3
Which Fields Should You Index?	9-4
Unique Fields	9-4
Fields Used to Retrieve Rows	9-4
Fields You Sort By	9-5
Default Structure Index Maps	9-5
Defining Additional Fields to Index	9-5
Customized Structure Index Maps.	9-7
Populating an Index Map.	9-8
 10 Data Conversion and Validation	
Overview of Data Conversion and Validation.	10-2
Data Conversion	10-2
Data Validation	10-2

Fields Can Have Internal and External Values	10-2
Automatically Generated Conversion and Validation Code	10-3
Field Conversion Code	10-4
Field Validation Code	10-7
Variables for Field Conversion and Validation Code	10-9
Order of Field Conversion and Validation Events	10-10
Modifying Field Conversion and Validation Code	10-11
Base Table Validation Code	10-15
Defining Base Table Validation Code	10-16
11 Base Table Triggers	
Overview of Open M with SQL Triggers.	11-2
You Can Associate Triggers with Different Kinds of Objects	11-2
Overview of Base Table Triggers.	11-4
Pre-Filing Triggers.	11-4
Post-Filing Triggers	11-5
SQL Code Triggers	11-5
M Code Triggers	11-6
M Routine Triggers	11-6
Base Table Trigger Code	11-7
Trigger Items	11-7
Trigger Action Types.	11-7
Conditional Execution of Triggers	11-7
Execution of Multiple Trigger Items	11-8
You May Reference Field Values in Base Table Triggers	11-8
%-Variables in Base Table Triggers	11-8
Defining a Base Table Trigger	11-10
Automatic Trigger Deletes Child Table Rows	11-14
Examples of Base Table Triggers.	11-15
12 Base Table Help and Error Messages	
Base Table Access Privileges.	12-2
Base Table Advanced Options	12-3
Help Text	12-5
Row Insert and Delete Messages	12-7
Developer Comments	12-11
Comment Reports	12-12

13 Field Help and Error Messages

Overview of the Field Help Facility	13-2
M with SQL Help Menu	13-2
Three Types of Field Help	13-2
Error Messages	13-2
Users Press <Explain> to Display Help	13-2
Override Help	13-3
Choosing Between Long Help and Override Help	13-3
Defining Long Help and Error Messages	13-4
Translations	13-4
Defining Override Help	13-8

14 Views

What Is a View?	14-2
How to Use Views	14-3
Types of Views	14-4
All Views Based on SELECT Queries	14-4
Creating a View	14-5
Defining a Table-Based View	14-8
Starting Table Is the Table on Which View is Based	14-8
Join Specifications Link Base Tables	14-8
Examples of Join Specifications	14-10
Sub-Views Copy Join Specifications Automatically	14-11
You Must Designate Fields to Appear in the View	14-11
Defining a Query-Based View	14-17
Observe These Restrictions on Query-Based Views	14-17
Copy an Existing Query Into the Current View	14-18
Copy Another View's Query Into the Current View	14-19
You Can Convert a Query-Based View to Table-Based	14-20
You Cannot Convert a Table-Based View to Query-Based	14-20
View-Related DDL Statements for Query-based Views	14-20
View Definition Options Menu	14-21
View Definition Advanced Options	14-21
View Definition Reports	14-22
View Definition Comments	14-22
Editing a View	14-23
Deleting a View	14-24
Deleting a View Automatically Deletes Dependent Objects	14-24
Changing a View's Starting Table	14-25
Restrictions on the Use of This Utility	14-25

Copying a View	14-28
You Must Have Privileges to Copy a View.....	14-28

15 Data Dictionary Reports

How To Run a Data Dictionary Report	15-2
List of Base Tables Report.....	15-4
Base Table (Order by Field Name) Report.....	15-5
Base Table (Order by Column Number) Report	15-6
Series of Base Tables Report	15-7
List Table Relationships Report.....	15-8
List of Views Report	15-9
View -- Sorted by Path Report	15-9
View -- Sorted by Fields Report.....	15-11
Global Documentation Report	15-13
Global Doc -- Map Version Report	15-15

Part III—Physical Structure

16 Default Physical Structure

Overview of Default Physical Structure.....	16-2
Examining Default Physical Structure is Optional	16-2
Converting from Default to Custom Physical Structure	16-3
Examining the Row ID Field Definition	16-3
Row ID Definition in Non-Child Tables	16-4
Row ID Definition in Child Tables	16-5
Accessing the Master Map and Index Maps.....	16-10
Examining a Default Master Map	16-12
New and Old Master Map Structures	16-12
SET Commands for Master Map Structure	16-14
Map Definition Master Window	16-15
Access Path Specifications	16-18
Full Row Reference	16-19
Map Data Specifications	16-19
Row ID Specifications.....	16-20
Child Table Default Master Map.....	16-20
Examining a Default Index Map	16-22
Index Maps for Name Data Type Fields	16-24
Changing the Global Name in a Default Physical Structure	16-25

Changing a Global's System or Directory	16-26
Updating Default Physical Structure	16-27

17 Relational Definition of an M Database

Overview of Creating a Relational Definition.	17-2
Existing M Database	17-2
New M Database	17-2
Examples	17-2
Checklist for Creating a Relational Definition	17-4
Relational Definition Checklist	17-4
Step 1: Design a Relational Definition	17-5
Identify Potential Tables	17-5
Identify Characteristic Relationships	17-6
Identify Fields in Each Table	17-7
NULL Values in Fields	17-7
Identify the Row ID of Each Table	17-7
Complex Global Structures	17-9
Step 2: Create a Base Table	17-9
Define Child Tables	17-9
Define Fields Using Default Physical Structure	17-9
Step 3: Edit Row ID Field(s)	17-10
Change to Customized Structure Before Editing Row ID Field and Mapping 17-10	
Default Row ID Field in Non-Child Tables	17-10
Default Row ID Fields in Child Tables	17-11
Row ID Field is a Single Data Field	17-11
Full Row is a Delimited Piece of a Global Node	17-12
Step 4: Define One Data Field	17-13
Step 5: Create Preliminary Master Map	17-13
You Must Define Row ID	17-13
A Master Map Cannot Be a Conditional Map	17-13
How to Create a Preliminary Master Map	17-13
Step 6: Test Access Path Specifications	17-14
Step 7: Define Remaining Base Table Fields	17-16
Some Fields You Define May Not Be in M Database	17-16
Step 8: Complete Master Map Definition	17-16
Step 9: Create Index Maps	17-16
Row ID Must Exist in Global Index Structure	17-16
Step 10: Create Conditional Maps	17-17
Step 11: Test Your Map Definitions	17-17

Step 12: Complete Base Table Definition	17-19
Example Using Enhanced ^SAMPLE Global	17-19
Description of Enhancement	17-19
Identify Potential Tables	17-20
Identify Fields in Each Table	17-20
Identify the Row ID of Each Table	17-20
Create a Base Table	17-20
Edit Row ID Field	17-21
Define One Data Field	17-21
Create Master Map	17-21
Example Using ^ACCT Global	17-22
Description of ^ACCT Global	17-22
Identify Potential Tables	17-23
Identify Characteristic Relationships	17-24
Identify Fields in Each Table	17-24
NULL Values in Fields	17-26
Identify Row ID of Table	17-26
Edit Row ID Fields	17-27
Example Using ^FLAVORS Global	17-29
Description of ^FLAVORS Global	17-29
Identify Fields and Tables	17-29
Identify Characteristic Relationships	17-30
Identify Fields in Each Table	17-31
Identify Row ID of Table	17-31
Edit the Row ID Field(s)	17-32

18 Creating a Customized Map Definition

Overview of Customized Map Definition	18-2
Sample Globals and Base Tables	18-2
Map Definition Checklist	18-3
Field Names for Map Definition	18-4
Step 1: Compose SET Command to Insert Row	18-5
Step 2: Enter Map Definition Form	18-6
If You Started Using Customized Physical Structure	18-6
If You Started Using Default Physical Structure	18-7
Examining the Map Definition Master Window	18-8
Step 3: Specify Global Name	18-11
Data Stored in One Global	18-11
Extended Global Reference	18-11
Data Stored in Two or More Globals	18-12
Data Stored in Local Array	18-12

Step 4: Define Standard Access Path Specifications	18-14
A. Enter Access Path Specifications Form	18-15
B. Examine, and If Necessary, Override, Data Access Expression . .	18-17
C. Specify How to Reach Current Access Level	18-20
Examples of Access Path Specifications	18-26
Delete Access Level Specifications	18-27
Step 5: Specify Special Access Code	18-29
How Open M with SQL Accesses and Validates a Row	18-29
Define Special Access Code to Validate Row	18-30
Types of Special Access Code	18-31
Range of Subscript Values Not Part of Base Table	18-32
Single Value Access and Invalid Values	18-33
NEXT Subroutine to Access Valid Access Level Values.	18-34
Example of Using Special Code	18-36
Step 6: Specify Additional Data Access Variables	18-39
Names of Additional Data Access Variables	18-39
Where You Can Use Data Access Variables	18-39
Step 7: Specify Override to Full Row Reference	18-41
Default Full Row Reference	18-41
{ %row } Represents Full Row Reference	18-41
Override Full Row Reference	18-41
Step 8: Edit Row ID Specifications	18-43
Step 9: Define Map Data Specifications	18-45
Two Ways to Define Field's Location	18-45
Standard Specifications	18-45
Examples of Standard Map Data Specifications	18-47
Retrieval M Code	18-50
Examples of Map Data Specifications	18-52
Step 10: Define Override Filing Code	18-55
Two Ways to Define Override Filing Code	18-55
Referencing Fields in Filing Code	18-55
Step 11: Define an Index Map as a Conditional Map	18-58

Part IV—Special Topics

19 The FileMan Interface

Overview of the FileMan Interface	19-2
Database Normalization	19-2
Conversion Logic	19-2
Identifiers	19-3
FileMan Preferences	19-4

File Specifications	19-5
Field Specifications	19-6
Indices	19-7
FileMan Interface Does Not Link ScreenMan Applications	19-7
Accessing the FileMan Interface	19-8
Creating the FileMan-Open M Link.	19-10
Creating The Link for All FileMan Files Within a Range.	19-10
Creating The Link One File at a Time.	19-12
Extending, Updating, and Deleting the FileMan-Open M Link	19-13
Link Maintenance Facility Runs Many Operations in Succession	19-13
Accessing the Link Maintenance Facility	19-13
Extending the FileMan-Open M with SQL Link.	19-15
Updating the FileMan-Open M with SQL Link	19-18
Deleting the FileMan-Open M with SQL Link	19-20
Generating Views of FileMan Files	19-23
FileMan-Open M with SQL Cross Reference Report	19-25
Conversion Error Log.	19-27
Print Conversion Error Log Report	19-27
Purge Conversion Error Log	19-28
Using the FileMan Interface with Open M with SQL PDP	19-29
Limitations to the FileMan Interface	19-30
FileMan Interface Questions and Answers.	19-32

20 Importing Data Definitions

Overview of Data Dictionary Import	20-2
Create ^mxdd Global	20-2
Basic Structure of ^mxdd Import Global	20-2
Provide Definition as Described in Data Dictionary	20-2
Values Provided as Subscripts and Node Values	20-2
Abbreviations Used to Represent Values	20-3
Fixed Subscript Values	20-3
Identify Object as Base Table or View	20-3
Required and Default Values.	20-3
Multi-Line Node Structure.	20-4
Base Table Node Structure.	20-5
Base Table Basic Definition	20-5
Base Table Field Definitions	20-7
Base Table Field Error Translations	20-13
Base Table Field Help Translations.	20-14
Base Table Triggers Definition	20-15
Base Table Map Node Structure	20-17

Base Table Map Access Path Specifications Structure.	20-18
Base Table Map Data Access Variable.	20-21
Base Table Row ID Specifications	20-22
Base Table Map Data Specifications Structure.	20-23
Default Physical Structure Index Definitions	20-23
View Node Structure	20-25
View Basic Definition.	20-25
View Join Specification	20-26
View Field Definition	20-27
Privileges Definition Node Structure.	20-30

A Data Dictionary Specifications

Data Dictionary Specifications	A-1
--	-----

B Keyboard Actions

Altos	B-1
ANSI	B-3
CIT-500.	B-4
COBRA.	B-6
DECTERM	B-7
DTM-PC Console	B-9
DESQView Console	B-9
ED3638	B-9
FALCO	B-11
Generic	B-12
IBM 3151-ANSI	B-14
IBM 3151 Ascii Display	B-15
IBM 6091	B-17
IBM PC	B-18
IBM PC With Color	B-20
MSM PC Console	B-22
NT Console	B-23
Open M Terminal	B-25
PC Console	B-26
SUN.	B-28
TV905	B-29
Unisys Console	B-31
VT100	B-32
VT220	B-34
WAYTEC	B-35
WYSE60 (Native).	B-37
WYSE85	B-39

Xterm..... B-40

- C SQL Reserved Words
- D Base Table Design Charts
- Glossary of Terms

Introduction

Chapter 1

Introduction to the Data Dictionary

Chapter 2

The Open M with SQL Relational Database

Chapter 3

Using the Data Dictionary Interface

Chapter 4

Designing a Sample Application

Introduction to the Data Dictionary

InterSystems' Open M with SQL Data Dictionary is the foundation upon which an Open M with SQL relational database application is built.

This chapter provides an overview of the Open M with SQL application development environment, focusing particular attention on the role played by the Data Dictionary.

Specifically, it covers the following topics:

- n The Open M with SQL Application Development Environment page 1-2
- n Defining a Relational Database in the Data Dictionary page 1-7
- n Mapping a Relational Database to M Globals page 1-13
- n The Data Dictionary Provides Many Benefits page 1-15

The Open M with SQL Application Development Environment

Open M with SQL is an integrated environment for developing and running database applications. It includes an advanced relational database management system, an application and report generator, and a procedural programming language.

Open M with SQL combines two ANSI-standard languages — M, the only procedural programming language designed specifically for database applications, and SQL (Structured Query Language), the most widely used relational query language.

Open M with SQL provides two program development environments:

- The M environment, in which you can execute system management commands, and create, modify, and execute M routines.
- The relational environment, in which you define the database in relational terms and use the application generator tools to create, modify, and execute advanced relational database applications.

Open M with SQL's relational environment is a complete RDBMS (Relational Database Management System)—it includes a development environment for creating advanced relational database applications, a management system for maintaining them, and a run-time environment for executing them.

Defining a Relational Database in the Data Dictionary

You define a relational database using the Data Dictionary.

You do this by defining tables to represent the logical structure of a relational database, or by mapping the physical structure of an existing database to relational tables. These tables act as the foundation on which Open M with SQL relational database applications are built.

Open M with SQL implements your database on the physical level as an M global database. If you already have an existing M database, you can define a relational “view” of it in the Data Dictionary. You do not need to know anything about M programming to design a relational database in the Data Dictionary.

Defining a relational database using the Open M with SQL Data Dictionary allows you to take advantage of the programming ease and efficient I/O of M while gaining centralization of data control and the data access power provided by SQL.

Open M with SQL Combines Two ANSI-Standard Languages: SQL and M

Open M with SQL combines two ANSI-Standard programming languages, SQL and M. The merger of these two languages provides a powerful standards-based system that offers extremely high performance.

Although you can develop fully functional relational applications without writing any code yourself, you also can take advantage of these two ANSI-Standard languages to customize generated applications to your exact specifications.

SQL

SQL is the most widely used relational query language. SQL has several component parts, including DDL, a data definition language, and DML, a data manipulation language. Open M with SQL primarily implements the DML component of SQL, although it also implements a small portion of the DDL component.

In Open M with SQL, you can use SQL to query the database, or to insert, update, or delete database rows. You can create and run SQL queries using the Query Generator or the Interactive Query Editor, or by embedding SQL queries within M macro source routines. Open M with SQL uses SQL internally to access data in Form Generator forms and M/PACT reports. SQL respects all the data constraints you define in the Data Dictionary.

InterSystems has implemented a number of extensions to SQL. These extensions include outer joins and extended arrow syntax to refer to implicit joins, both of which are discussed in this guide. For information on all InterSystems extensions to SQL, see the *Open M/SQL Developer Guide*.

M Language

M is a database-oriented procedural programming language for developing and running performance-critical online applications.

In M, data stored on disk are called globals. You can think of a global as a permanent variable. You refer to a global by its name; for instance, you might give a global a value by issuing the command, SET ^name="Jane Doe".

You can make direct global references to the M globals you define relationally in the Data Dictionary, as well as to M globals you have not defined relationally. However, when you access data directly with M global references, Open M does not respect the data constraints you define in the Data Dictionary. For this reason, InterSystems recommends that you do not use M global references to access your relationally described data, except during the conversion of a non-relational M application to an Open M with SQL relational application.

The Open M product family encompasses several dialects of M—ISM, DTM, and DSM. Each of these dialects is tailored to a particular hardware platform.

For more information about the M language, see the *M Language Reference*, which is a comprehensive reference manual for both the ISM and DTM implementations of M.

Open M Developer

Open M Developer is InterSystems' application development environment for creating highly sophisticated relational database applications.

Open M Developer has the following components:

- ▮ **Form Generator** — an application generation tool used to create ready-to-run data screens through which end users can add, retrieve, edit, and delete database information.

See the *Open M/SQL User Interface Programming Guide* for complete information on creating forms.
- ▮ **Menu Generator** — an application generation tool used to create pop-up menus and menu bars which unite the various components of an application in a logical and visually sophisticated manner and structure an application by defining how it is organized and how it is presented to end users.
- ▮ **Query Generator** — environment in which you can create and run database queries on an ad-hoc basis.
- ▮ **M/PACT (Report Generator)** — a report-writing tool used to create and run sophisticated end-user data reporting applications that interact with the Data Dictionary.

Note: M/PACT is an add-on option.

See the *Open M/SQL Developer Guide* and *Open M/SQL User Interface Programming Guide* for information on using the Open M with SQL application development environment.

Open M with SQL Server Opens Your Relational Database

The Open M Relational Server is an add-on component to the Open M Developer product that allows non-M applications to access and modify your Open M with SQL relational database.

The Relational Server opens your data to new computing technologies by allowing diverse applications to access your Open M with SQL relational database.

The Relational Server is designed to let you retrieve and modify M data using the following application development tools:

- n Software packages such as spreadsheets, graphical user interface development tools, computer aided software engineering (CASE) tools, natural language data access interfaces, and executive information systems.
- n Applications developed in the C programming language.

For information on how to configure, manage, and maintain the Relational Server, see the *Open M with SQL Database Administrator's Guide*.

For information on how to access an Open M with SQL relational database (retrieve and modify data) from various third-party application development tools, see the *Open M/SQL Server User Guide*.

For information on how to access an Open M with SQL relational database from applications created using the C or C++ programming language, see the *Open M/SQL Server Programming Guide*.

How is Open M with SQL “Open”

The “Open” in Open M with SQL refers to its open systems architecture, which makes applications easily portable and opens data to a wide variety of computing technologies.

Specifically, Open M with SQL is open in the following ways:

- n All Open M with SQL applications are completely portable across M implementations — you can develop Open M with SQL applications on top of any vendors’ M system and then run the application on the same M system or on any other M system.

For example, you can develop an Open M with SQL application on top of Micronetics’ MSM and then run the application on MSM or on any other M system.

- n All Open M with SQL applications are completely portable across hardware platforms — you can develop applications on one hardware platform and then run them on another.

For example, you might develop an Open M with SQL application on a VAX DSM system, and then recompile and run the application on a SCO workstation.

- n You may use the Open M Relational Server to access data stored in an Open M with SQL relational database from various third-party software applications as well as from applications developed in the C programming language.
- n You may access data in your Open M with SQL relational database using a variety of distributed data processing protocols, including InterSystems' proprietary ISNET protocol over Ethernet or TCP systems as well as DSM-DDP and OMI.

Developing Relational Applications

Open M with SQL offers a variety of strategies for developing relational applications:

- n Automated application generation — using the Data Dictionary, Form Generator, M/PACT, and Menu Generator
- n Hand-coded programming — using M and/or SQL
- n Mixed strategy

Most commonly, applications are developed in a mixed environment that includes programs created using M as well as automatically-generated forms, reports, and SQL queries created using the Open M with SQL application generation tools. It generally makes sense to use the application generator tools to develop as much of the application as possible, then to use hand-coded programming to articulate the refinements and customizations that even the advanced customization features do not handle.

For example, a laboratory application might perform most of its data entry through data entry screens created using the Form Generator, but it may use hand-coded M routines for direct data input from specialized instruments. The result is an integrated application created and run entirely within Open M with SQL.

For more information on application development strategies, see the *Open M/SQL Developer Guide*.

Defining a Relational Database in the Data Dictionary

The Data Dictionary is the foundation of the Open M with SQL application development environment. The Data Dictionary provides an advanced, windows-based environment in which you define the elements of a relational database.

A relational database in Open M with SQL is a functionally-related group of base tables which are explicitly linked together in the Data Dictionary. You can join base tables within a single database to create views. Views are considered virtual tables because they appear as tables to the end user but are not stored as such in the database.

Data Dictionary maps define the connection between the logical structure and physical storage structure of a database. The Data Dictionary can automatically generate maps to create a default physical structure for your database. Alternatively, advanced M programmers may wish to define their own maps to conform with the global structures of existing applications or to design a new database structure compliant with application specifications.

Once you have defined a relational database in the Data Dictionary, you can use the other components of the Open M with SQL application development environment—the Form Generator, Menu Generator, Application Help Facility, SQL query facilities, and the M/PACT Report Generator—to quickly and easily build a complete relational database application.

Although describing the database in the Data Dictionary is essential to the use of the Form Generator, the SQL query facilities, and M/PACT, it is not required for M programs that access the database only through direct global references.

Mapping the Functional Specifications of Your Database

The first step in creating a database is to organize the logical structure of the data on paper as a series of tables, just as you might design a paper filing system. To do this, consider the functional needs of your application and make a list of all the different pieces of data that you need. The object of the design is to describe the tables that constitute the database and how those tables will interact with one another. For example, one database application might have a customer table, an invoice table, an invoice line item table, a parts table, etc. The customer table might then consist of fields for customer name, address, phone, current balance, etc.

Defining the Data Structure in the Data Dictionary

Once you have designed a data structure on paper, you next define the data structure in the Data Dictionary, along with integrity constraints on the data.

Integrity constraints may be very simple; for example, you can specify a range of valid numbers that can be entered for a field. Other constraints may be more complex; perhaps you want to specify that an invoice cannot be entered into an invoice table without a corresponding customer entry in the customer table. Even more complex constraints may require the addition of complete SQL queries or procedural M code.

You describe relationships among tables in the Data Dictionary by defining **characteristic** or **designative** links between tables. For example, where rows of an invoice line item table cannot exist without a corresponding row in an invoice table, you should define the line item table as a characteristic table of the invoice table. You might also define one field in the invoice table as a **designative reference** to rows in the customer table. When you define these relationships, the Data Dictionary creates “implicit” joins between the tables. These joins are automatically generated in queries, freeing the programmer and end-user from the tedious and demanding task of specifying explicit joins in SQL statements.

You may also define processing triggers in the Data Dictionary. Triggers are sequences of actions defined to automatically occur, or be *triggered*, when certain other events occur. A trigger definition usually consists of an SQL query or lines of M code that are invoked at row insert, update, or delete time. For example, if a medical record for a given patient is deleted, you may want to establish a trigger that will automatically delete all of that patient's lab test information from various lab files.

Defining Existing M Databases Relationally

You can use the Data Dictionary to link the physical storage of an existing non-relational M database to the logical structure of the relational tables you define in the Data Dictionary. For more information about converting your existing databases, see the section of this chapter entitled “Mapping a Relational Database to M Globals” on page 1-13.

Once you provide a relational picture of your M database in the Data Dictionary, you can access it using both SQL and direct M global references. In this way existing M applications can incorporate the use of SQL over time; immediate conversion is not required.

Accessing Data

You can choose to access data in the relational database in the following ways:

- n Form Generator forms
- n M/PACT reports
- n SQL queries
 - Defined via the Query Generator or Interactive Query Editor
 - Embedded in M macro source routines
 - In triggers

Note: Triggers are special actions you define to occur under certain circumstances, such as when a user enters a particular value in a field on a Form Generator form, saves new data in a base table, or selects a particular option on a menu

- n Direct global references in M routines

Note: The data constraints you define in the Data Dictionary are not respected when you access data using M global reference.

Defining Base Tables in the Data Dictionary

The **base table** is the fundamental structure in an Open M with SQL relational database. Most of your work in the Data Dictionary will be spent creating and defining base tables.

For every base table you define in the Data Dictionary, you can specify:

- n How you will define the underlying M global structure that the base table represents:
 - Using default physical structure
 - Using your own customized physical structure
- n The name of the M global or globals that contain the data defined in the base table.
- n The lookup fields into which users can enter data on a Form Generator form to retrieve a row from a base table.
- n Actions, called triggers, that occur at row insert, update, or delete time.
- n Other base tables that contain data related to the data this table—these other base tables are associated by characteristic relationships or designative references.

See Chapter 5, *Defining a Base Table*, for complete information on how you define a base table.

Characteristic Relationships Join Base Tables

Your database may include a one-to-many (parent-to-child) data relationship, where many subsets of data are associated with one particular piece of data.

For instance, perhaps you have a base table *Invoices*, which includes invoice number, customer name, customer address, etc. Each invoice has many associated line items. Each line item includes item number and name, quantity purchased, purchase price, etc. It is easier and more efficient to store line item data in a separate base table.

In order to link each line item—that is, each row—in the *LineItem* base table to an invoice, or row, in the *Invoices* base table, you define a characteristic relationship between the two tables. To do so, when you define the *Invoices* base table, you specify the *LineItem* base table as a child table of *Invoices*.

Once you have defined a characteristic relationship between tables, you can use SQL (taking advantage of InterSystems' extensions) to access data in child and parent tables. You can also create forms that allow users to enter data into child tables while entering data into the parent table.

For information on how to define characteristic relationships between base tables, see Chapter 5, *Defining a Base Table*.

For a complete description of characteristic relationships and the way Open M with SQL implements them as implicit joins, see Chapter 7, *Implicit Joins*.

Designative Reference Fields Join Base Tables

Sometimes, data in one table has a many-to-one relationship with data in another table. To represent this type of relationship, you can define field with the special data type, Designative Reference. Designative Reference fields associate a row of data in one base table with a row of data in another base table.

For example, you may have two base tables: *Patients* and *Doctors*. You want to link each row—that is, each patient—in the *Patient* table to a specific row—a doctor—in the *Doctors* table. Many patients may be linked to the same doctor (many-to-one).

You create this link by defining a Designative Reference field in the *Patients* base table which points to the *Doctors* base table.

Once you have defined a Designative Reference field, you can use SQL (taking advantage of InterSystems' extensions) to access data in the designated table.

When using the Form Generator to create forms based on a table that has a Designative Reference field, you may create multiple Designative Display fields—as many as one for each field in the designated table. These Designative Display fields allow you to display data for the corresponding fields in the designated table.

For example, when creating a Patients data entry form, you may create Designative Display fields for DoctorName and DoctorPhone. Users run the form and enter data on a new patient. When they reach the DoctorName field, they may select a doctor from the doctors in the Doctors table. When they select a particular doctor, that doctor's phone number is displayed in the DoctorPhone field.

For information on defining Designative Reference fields, see Chapter 6, *Defining Base Table Fields*.

For a complete description of designative relationships and the way Open M with SQL implements them as implicit joins, see Chapter 7, *Implicit Joins*.

For information on creating Designative Display fields for forms, see the *Open M/SQL User Interface Programming Guide*.

Defining Database Fields in Base Tables

The fields you create when defining base tables are called database fields. Database fields refer to M globals. When you create forms using the Form Generator, you associate each form with a particular base table, and you may display any database field defined for the base table on the form. Users run these forms to insert, edit and delete data in the database.

Note You can also create form-only fields in the Form Generator. These fields do not relate to database fields defined in the Data Dictionary and are not stored in the database.

You define default characteristics for database fields in the Data Dictionary. You may override some of these characteristics at the form level in the Form Generator.

Defining Attributes of Database Fields

The Data Dictionary allows you to define the following attributes for each field in a base table:

- n Data type (including Designative Reference)
- n Descriptive caption for the field that displays in Form Generator forms and M/PACT reports
- n Maximum length of the field
- n Allowable range
- n Default value for the field (when a new row is inserted into the base table)
- n Possible values for the field (via Multiple Choice data type)
- n Single-line or multi-line
- n External-to-Internal conversion rules for the internal storage of data
- n Internal-to-External conversion rules for external data display
- n Rules for validating the field's value
- n M code lines to compute the value of the field based on other fields
- n Help and error messages that display when editing the field's value in a Form Generator form

Defining Views in the Data Dictionary

Once you have created base tables in the Data Dictionary, you can create **views**. Typically, views link together multiple tables within a single database, though you can also create views on a single table. With a view, users can see data for all (or a subset of all) the fields defined in the base table(s) which are part of the view.

Views are considered **virtual tables** because they appear to be tables to the end user but are not stored as such in the database. A view can be used as a data source in SQL queries and M/PACT reports. This allows you to include fields from more than one base table in your report or query.

Views also perform a security function by restricting users' access to specified fields in the viewed tables.

Views can be based on any of the following:

- n A base table
- n Another view
- n A query — such views are called General Views

For complete information on views and how to define them, see Chapter 14, *Views*.

Mapping a Relational Database to M Globals

Mapping is the process by which the Data Dictionary links the logical data structures you describe relationally in base tables to physical M globals. The Data Dictionary can automate this process so that it happens transparently, or you may choose to control the mapping yourself.

When the Data Dictionary maps to a global, that global need not exist. If it exists, it need not contain data.

Each base table has a Master Map, which describes the physical storage of its fields in M globals. It also may have one or more index maps, which enable the SQL query processor to locate particular rows in the base table more quickly. Base tables do not share maps. Two or more base tables can have fields linked to the same physical global, but different maps describe their storage.

Selecting Default or Customized Physical Structure

A major strength of the Data Dictionary is that, for each base table, you can choose among three mapping options:

n **Automatic Default-Generated Physical Structure**

The Data Dictionary automatically generates a default physical structure.

n **Hand-mapped Customized Physical Structure**

You specify the M global structure for the base tables you define.

n **Mixed Strategy**

The Data Dictionary automatically generates a default physical structure. It sets up the fields in the Master Map and defines an index for all lookup fields and all unique fields. You then remove all field links to globals and replace them with global references that reflect your M database.

When to Use Default Physical Structure

The default physical structure is an optimized map definition based on base table and field definitions that the Data Dictionary generates automatically. Using this option, you can develop an entire database application and never examine the map definitions or global data structures. If you are creating a new database application (not converting an existing M database), you may find it quick and easy to use the default physical structure, though some programmers will still prefer to define the mapping themselves.

Using the default physical structure has the following benefits:

- You need never enter the map definition area of the Data Dictionary or be concerned with the underlying physical structure of your base tables. This is desirable for programmers who intend to develop an entire database application using the Open M with SQL automatic application generation facilities. It is also desirable for programmers with little or no programming experience in M.
- Default structure automatically creates index maps for
 - Each field you define as a lookup field (lookup fields are used for looking up database rows in Form Generator forms)
 - Each field you define as a unique field
 - Each field you specify as an Index Field

The Data Dictionary automatically creates a default physical structure for your base table, unless you specify otherwise.

When to Use Customized Physical Structure

If you are converting an existing M database to an Open M with SQL relational database, you must do your own mapping. This allows you to link your relational database structure (base tables) to your existing M globals, a mapping which only you can specify. Once you have done this, you can use all Open M with SQL application generation facilities to build a complete database application for your existing M database.

Some M programmers prefer to do customized mapping for every base table they create. They feel this enables them to exercise more precise control over the internal processing of the application and to better optimize the application code. It can be a tedious process and it requires a thorough understanding of a complex environment, but if done properly, customized mapping can improve the processing efficiency of your database application.

Caution Once you choose to do customized mapping for a base table, you can never go back to the default physical structure. This means you must always update the physical structure to reflect any changes you make to logical structure.

The Data Dictionary Provides Many Benefits

The Data Dictionary provides all the following benefits:

- n Ability to create a relational database without designing the underlying physical structure
- n Ability to create a database structure that you can use as the foundation for building an entire relational database application, using the Open M with SQL application generator tools
- n Access to SQL
- n Enhanced developer productivity
- n Enhanced application performance via:
 - Query Optimizer, which uses sophisticated algorithms to convert SQL queries into highly efficient M code
 - Index maps, which enable the Query Optimizer to quickly locate particular rows in a base table
 - Continued use of existing M databases
 - Automatic data validation and automatic updates to other base tables via triggers, enforcing referential integrity constraints

Some of these benefits are discussed further in the following sections.

Developer Productivity

Whenever you use an SQL query to retrieve, insert, update, or delete rows in a base table, the Query Optimizer automatically generates all of the procedural code needed to perform the operation. The automatic generation of highly efficient M code routines can replace coding efforts sometimes taking months or years to complete manually. This code generation frees application developers to spend their time on other tasks, greatly increasing developer productivity.

Existing Applications

If you have an existing M database, the Data Dictionary lets you use customized mapping to link the relational database structure (base tables) you define in the Data Dictionary to one or more of your existing M globals. Any M database structure can be represented in the Data Dictionary. You can also use the Data Dictionary to enhance an existing global structure; for example, you can add index maps, which facilitates data retrieval. Thus, the Data Dictionary enables you to protect your current investment while enhancing your existing applications.

Referential Integrity

A database contains data which is logically related. It is important to make sure that applications can maintain the logical relationships between data elements during data entry and update. For example, an application should not allow a new employee to be assigned to a department that does not exist in the Departments table. Such requirements are known as referential integrity.

By defining your data in the Data Dictionary you can easily implement the data control functions necessary to most applications. The Data Dictionary provides a combination of automatic and user-defined features for preserving referential integrity in your database applications.

For example, you can:

- Define **triggers** that respond to the insert, update, and deletion of rows in a given table by automatically inserting, updating, or deleting rows in other tables. See Chapter 11, *Base Table Triggers*.
- Define **base table validation code** that prohibits filing of rows that violate a specified set of constraints among fields. Chapter 10, *Data Conversion and Validation*.
- Define **field validation code** that prohibits filing of values that do not satisfy a specifies set of conditions. See Chapter 10, *Data Conversion and Validation*.

You may implement additional referential integrity rules when designing forms using the Form Generator. For more information on defining form-level application integrity code, see the *Open M/SQL User Interface Programming Guide*.

The Open M with SQL Relational Database

Open M with SQL is based on an advanced relational model, which defines the database as a group of tables. InterSystems has extended the relational model to allow the definition of designative and characteristic relationships between tables. In Open M with SQL, any related group of tables that resides within an M data partition is considered a database. The M data partition is usually referred to as a directory or UCI, depending on your M system. An Open M with SQL database corresponds to the *schema* in the relational model.

Open M with SQL considers any related group of tables that resides within an M data partition to be a database. The M data partition is usually referred to as a directory or UCI, depending on your M system.

This chapter discusses relational concepts and their implementation in Open M with SQL.

Specifically, it covers the following topics:

n	Relational Database Is a Collection of Tables	page 2-2
n	Open M with SQL Supports Two Types of Tables	page 2-3
n	Open M with SQL Tables Follow First Normal Form	page 2-6
n	Open M with SQL Implements Schemas as M Directories	page 2-8
n	Accessing Data in an Open M with SQL Relational Database	page 2-9
n	InterSystems' Extensions to the Relational Model	page 2-16

Relational Database Is a Collection of Tables

A relational database is characterized by:

- n A database structure that consists of a collection of **tables** (also called “relations” or “files”).
- n The presence of a query language that permits the manipulation of these tables in a **mathematically complete** manner.

The table is the basic unit of data storage in the relational model. A table is a collection of **rows** (also called “tuples” or “records”) and **columns** (also called “fields” or “attributes”). Each column in a table contains a particular type of data, such as integer, date, or text. (Columns may also contain null values.) Each row in a table corresponds to a real world entity and contains exactly one value for each column in the table.

Example Below is a sample relational table called “Employees”. Each row of this table refers to one entity—in this case, an employee. Each row contains a value in each column, all referring to the same employee.

Table 2-1: A Sample Relational Table

EmpName	EmpNum	HireDate	Status	Salary
Grainger,Lisa	445-67-7891	06/06/90	Part-Time	28,000.00
Corson,Bob	210-92-8518	12/08/86	Active	15,000.00
Doe,Suzanne	333-44-7800	01/05/87	Active	38,000.00
Fast,Felix	334-45-5678	06/23/89	Active	44,000.00
Finley,Jack	356-62-1221	09/09/85	Part-Time	77,500.00
Gable,Bill	567-89-0123	08/02/92	Active	44,000.00

Open M with SQL Supports Two Types of Tables

Open M with SQL supports two types of tables:

- n Base tables
- n Virtual tables

A base table is an autonomous, named table. Unlike virtual tables, base tables exist physically in the sense that they are mapped directly to physical storage structures. We say that base tables are “named” because the table is explicitly given a name via an appropriate definition statement, unlike, for example, the result of a query, which is not explicitly named and exists only ephemerally.

A virtual table is a named table derived from one or more base tables. Virtual tables are not directly represented in physical storage. Rather, they are abstract collections of base tables.

Examples of virtual tables include the output from SELECT-statement queries and **views**. Views are windows through which data from multiple base tables can be “viewed”. Open M with SQL allows views to serve as data sources for M/PACT reports, SQL queries, and other views.

No two rows of a base table are identical. In virtual tables, two or more rows may be identical.

Row ID/Primary Key

The **primary key** or **Row ID** is a field or combination of fields that serve as the unique identifier to each row in a base table. At any given time, no two rows of the base table may contain the same primary key value. For example, the primary key of an “Employees” table might be the unique employee number; as each employee is identified by a unique employee number, each employee row in the table is identified by the value of the employee number field.

Since two rows of a base table cannot be identical while two rows of a virtual table can, only base tables have primary keys. Virtual tables (results of queries, for example) do not have primary keys.

In Open M with SQL, the Row ID field is the primary key. The Row ID is a single field of the table that uniquely identifies the row. Open M with SQL automatically creates a field of data type Row ID for every table, whether you use default or customized physical structure. To learn more about the default-generated Row ID field, see Chapter 16, *Default Physical Structure* .

When you use customized physical structure, you usually need to edit the Row ID field. You can either define a single field, which contains sequential integer values or a compound field based on one or more existing fields in your base table. You must be certain that, taken together, these fields uniquely define a row.

For example, you might define the Row ID of the Employees table to be the combination of the fields EmpName and EmpNum. To learn more about defining the Row ID field when using customized physical structure, refer to “Step 3: Edit Row ID Field(s)” on page 17-10 in Chapter 17, *Relational Definition of an M Database*.

Base Tables Contain Data Stored on Disk

The base table is the fundamental unit of data storage in the Open M with SQL relational database. You define base tables in the Data Dictionary. Base tables contain the database fields which hold the data for your application. Base tables are mapped directly to your M global database. The data stored in a base table is permanent, because it is stored on disk.

Virtual Tables Exist in Temporary Memory

Virtual tables are transient tables which exist only temporarily and have no permanent physical storage.

Virtual tables are derived from SQL queries. They represent a subset of data from a particular base table or from a combination of several base tables. Unlike base tables, virtual tables may contain duplicate rows.

Open M with SQL supports two types of virtual tables:

- Query output
- Views

An SQL Query Produces a Virtual Table

The output returned by an SQL SELECT-statement query constitutes a virtual table. In Open M with SQL, you may generate SELECT-statement queries in any of the following three ways:

- n Via the Query Definition template
- n Via the Interactive Query Editor
- n By embedding the query within M macro source code

Example Suppose you want to produce a seniority report using the Employees table. The report should include only the fields EmpName and HireDate, sorted in descending order by HireDate. To do this, you would issue the following SQL statement:

```
SELECT EmpName, HireDate
FROM Employees
ORDER BY HireDate DESC
```

For more information about creating queries using embedded SQL, the Query Editor, or the Query Generator, see the *Open M/SQL Developer Guide*.

A View Produces a Virtual Table

Views are conceptual windows through which data from one or many base tables can be “viewed”. Open M with SQL allows views to serve as data sources for M/PACT reports, SQL queries, and other views.

Typically, you define views in the Data Dictionary.

Note You can also use the SQL DDL keyword CREATE VIEW to define views.

Open M with SQL supports three types of views:

- n A view based on a base table. These views can include fields from many base tables, as long as they have either a characteristic relationship or a designative relationship with the starting base table.
- n A view based on another view.
- n A view based on a query.

For more information about defining and using views, see Chapter 14, *Views*.

Benefits of Views

Views provide the following benefits:

- n You gain access to a cross-section of fields from multiple base tables with a simple SQL statement.
- n Since M/PACT can access only one base table per report, a view makes it possible to display fields from multiple base tables in a single M/PACT report.
- n As a form of data security, you can include only some fields from a base table in a view and give users access to the view rather than to the underlying base table.
- n Views simplify the conceptual relationships between base tables so that fields which may actually reside in many different tables can be viewed logically as belonging to the same row in one “super table”. Users often find this to be a much easier way to get information out of a database.

Open M with SQL Tables Follow First Normal Form

Tables in Open M with SQL follow what is called the **First Normal Form**; that is, each column of a single row contains exactly one value, which may be null if the field is empty.

In some non-relational database systems, a column for a single row might contain several values. For example, the “Child” field in a “People” table might contain the names of three children; a single child field might also have three grandchildren.

However, in a relational system, each parent/child/grandchild relationship is represented by a separate row. For instance, each child of a given parent will occupy a unique row in the table despite the commonality of their parent fields. Likewise, each grandchild of that parent will also occupy its own unique row. This means that if a parent has three children who in turn have three children apiece, the relational model uses nine separate rows to describe the entire genealogy.

Alternatively, you may use separate tables called characteristic tables (see below) to designate relationships between parents, children, and grandchildren. It is your responsibility to define how you want Open M with SQL to handle parent/child relationships.

Example In a non-relational database that does not follow the First Normal Form, suppose you have a table called “Family”. A “Parents” field contains the name of both parents, and a “Children” field contains the names of all children of those parents, as shown below:

Table 2-2: Sample Non-Relational Table Not Using First Normal Form

Parents	Children
John Doe Mary Doe	Joseph Jane Derek

Using a relational model that does follow the First Normal Form, you define separate fields for “Mother” and “Father”. Furthermore, each child of a given parent occupies a unique row in the table despite the commonality of their parent fields. Therefore, if parents have three children, the relational model uses three separate rows to describe the genealogy, as shown below:

Table 2-3: Sample Relational Table Using First Normal Form

MotherName	FatherName	Phone	City	Child
Mary	John	222-3333	Andover	Joseph
Mary	John	222-3333	Andover	Jane
Mary	John	222-3333	Andover	Derek

As you can see, this causes data redundancy, wasting disk space.

To avoid this data redundancy, Open M with SQL implements two special links, or implicit joins, between base tables:

- n Designative References
- n Characteristic Relationships

Instead of repeating data in multiple rows in the same table, you define two separate tables. A row in one table points to a row in another table.

When implemented using a characteristic relationship, the Family table becomes two separate tables — Parents and Children, as shown below:

Table 2-4: Parents Table

Row ID	Mother	Father	Phone	City
1	Mary	John	222-3333	Andover
2	Jane	Robert	333-4444	Dracut

Table 2-5: Children Table

Parent Row ID	Child Row ID	Name
1	1	Joseph
1	2	Jane
1	3	Derek
2	4	Jennifer

Each row in the “Parents” table points to zero, one, or many rows in the “Children” table. Each row in the “Children” table points back to one row in the “Parents” table.

For more information on implicit joins, see the section of this chapter entitled “InterSystems’ Extensions to the Relational Model” on page 2-16.

To learn how to define designative references and characteristic relationships, see Chapter 5, *Defining a Base Table*.

Open M with SQL Implements Schemas as M Directories

In the relational model, a schema is a conceptual repository for a group of relations. No relation within the schema can have the same name as another.

In Open M with SQL, the unit equivalent to the schema is an M database. In Open M with SQL, an M database can be spread over one or more directories. Each directory contains one component—OPENM.DAT file—of the entire M database. An OPENM.DAT file is usually referred to by the name of the operating system directory in which it resides. All tables defined in the same M directory must have unique names.

The system manager's directory contains a database directory table. The database directory table contains the database name, directory, and directory set if net-worked.

Open M with SQL Database Structure for Non-ISM Implementations of M

When Open M with SQL is layered on top of a non-ISM implementation of M, the database concept is implemented differently.

For example, under DSM, Open M with SQL considers all tables within a unique UCI and volume set to be a common database.

Other M systems employ various other schemes for storing the list of databases. For more information, see the documentation provided with your M system.

Accessing Data in an Open M with SQL Relational Database

Once you have defined tables in the Open M with SQL Relational Data Dictionary, you can access data from those tables by any of the following means:

- n SQL queries — you may embed queries in M macro source code, or you may define them via the Open M with SQL Query Definition template or the Interactive SQL Query Editor
- n M global references (in conjunction with or independent of SQL)
- n Data entry, inquiry, and update forms designed using the Form Generator
- n Reports defined and formatted using M/PACT

This section provides a brief overview of relational operations in SQL, the ANSI-Standard Query Language. SQL enables the retrieval of data from the relational database for the generation of queries and reports.

Refer to an SQL text for a full understanding of this language. The *Open M/SQL Developer Guide*, Chapter 9, “Open M/SQL Implementation of SQL”, describes InterSystems’ extensions to standard SQL.

Using SQL to Query the Database

Open M with SQL lets you query the database using standard SQL SELECT statement queries. Queries access data in tables and views. The output from a query forms a virtual table and leaves the targeted table(s) unchanged.

You can reference multiple tables in a single query.

SQL enables you to use the following relational operations when querying a database:

- n Combining tables (or, more accurately, viewing tables as if they had been combined into one larger table)
- n Selecting particular columns from a single table or a combined table
- n Specifying particular rows in a single table or a combined table

Relational algebra provides the conceptual foundation for these relational operations. Understanding how the relational algebra works will help you take full advantage of the capabilities of SQL.

Conceptually, the relational algebra operates on one or more tables to produce a new (virtual) table. For example, a join combines two tables into a third. Or, a select operation extracts selected rows from one table to produce another table. A virtual output table is the result of the relational operation.

The list of fundamental relational operators includes:

- n **Cartesian Product** — creates a cross-product of multiple tables, i.e., views the tables as if they had been combined into one larger table
- n **Project** — selects particular columns from a single table or a combined table
- n **Restrict** — selects particular rows from a single table or a combined table
- n **Join** — selects some fields from some rows of multiple tables based on some relationship between the fields of the different tables

Table.Name Syntax

When referencing fields in an SQL query you may optionally precede the field name with the name of the base table to which it belongs, using the following syntax:

```
Table.Field
```

For example, a SELECT clause might read as follows:

```
SELECT Employees.EmpName, Depts.DeptName
```

If you do not use prefixes when creating field names that reflect the name of the base table in which they reside, it is generally a good idea to use this syntax for clarity.

If a query references multiple tables and those tables contain fields with identical field names, you must use the Table.Name syntax.

Cartesian Product

The **Cartesian Product** of two tables is the cross-product of all possible combinations of rows from the two tables, such that each row of the first table is combined (concatenated) with each row of the second table.

In SQL, the Cartesian Product operation occurs when two or more tables are explicitly or implicitly (within a view) named in the FROM clause, all fields in those tables are named in the SELECT clause, and no WHERE clause is present to specify relationships among the tables.

Example In the example below, the tables “Employees” and “Departments” are combined to form a virtual table, which is their Cartesian Product, by issuing the following SQL statement:

```
SELECT *  
FROM Employees, Departments
```

The asterisk in the SQL SELECT statement is used to denote “all fields in the specified tables”.

The “Employees” table is shown below:

Table 2-6: Employees Table

EmpName	EmpNum	DeptNum
Bravo, Vicki	445-67-7800	1000
Doe, Suzanne	253-44-7898	3000
Corson, Bob	210-92-6518	2000

The “Departments” table is shown below:

Table 2-7: Departments Table

DeptNum	DeptName	NumEmp	DeptMgr
1000	Sales	50	Bravo, Vicki
2000	Administration	101	Corson, Bob
3000	Development	200	Doe, Suzanne

The table below represents the Cartesian Product of the “Employees” table and the “Departments” table:

Table 2-8: Output Table After Cartesian Product Operation

E.EmpName	E.EmpNum	E.Dept-Num	D.Dept-Num	D. DeptName	D.Num-Emp	D.DeptMgr
Bravo, Vicki	445-67-7800	1000	1000	Sales	50	Bravo,Vicki
Bravo, Vicki	445-67-7800	1000	2000	Administration	101	Corson, Bob
Bravo, Vicki	445-67-7800	1000	3000	Development	200	Doe, Suzanne
Corson, Bob	210-92-6518	2000	1000	Sales	50	Bravo,Vicki
Corson, Bob	210-92-6518	2000	2000	Administration	101	Corson, Bob
Corson, Bob	210-92-6518	2000	3000	Development	200	Doe, Suzanne
Doe, Suzanne	253-44-7898	3000	1000	Sales	50	Bravo,Vicki
Doe, Suzanne	253-44-7898	3000	2000	Administration	101	Corson, Bob
Doe, Suzanne	253-44-7898	3000	3000	Development	200	Doe, Suzanne

Note that this output may not be ordered by the “Employees.EmpName” field. The ordering is subject to the specifications in the ORDER BY clause of the SQL query, and

Project

The project operation extracts a subset of fields from an existing table. The result is a new table (a virtual table) with the same number of rows but fewer fields.

You can perform a Project operation on a single table or on a combination of tables.

Example In the example below, the fields “EmpName” and “DeptName” are selected from the Cartesian Product of the tables “Employees” and “Departments”.

```
SELECT Employees.EmpName, Departments.DeptName
FROM Employees, Departments
```

The SELECT clause specifies the fields to be included in the projected table. The Cartesian Product is derived from the tables named in the FROM clause.

This Project operation yields the following table:

Table 2-9: Output Table After Cartesian Product Operation

EmpName	DeptName
Bravo, Vicki	Sales
Bravo, Vicki	Administration
Bravo, Vicki	Development
Corson, Bob	Sales
Corson, Bob	Administration
Corson, Bob	Development
Doe, Suzanne	Sales
Doe, Suzanne	Administration
Doe, Suzanne	Development

Whether you perform a query on one or many tables, the conceptual result is always a new table that is derived from existing tables in the database.

Restrict

The Restrict operation selects a designated set of rows from one or more tables. In the SQL query language, restriction is expressed through the WHERE clause, which uses the comparison operations, such as >, <, and =, for example:

```
WHERE City="Boston"
```

or

```
WHERE Age>20
```

You can perform a Restrict operation on a single table or on a combination of tables.

Example In the example below, all fields are selected from the Cartesian Product of the tables “Employees” and “Departments”, but the output table is restricted to only those rows for which the “NumEmp” field has a value greater than 100:

```
SELECT *
FROM Employees,Departments
WHERE Employees.NumEmp > 100
```

The SELECT clause uses the asterisk (*) to select all fields for inclusion in the table. The Cartesian Product is derived from the tables named in the FROM clause. The WHERE clause designates the condition for the restriction operation.

This Restrict operation yields the following table:

Table 2-10: Output Table After Restrict Operation

E.EmpName	E.EmpNum	E.Dept-Num	D.Dept-Num	D.DeptName	D.Num-Emp	D.DeptMgr
Bravo, Vicki	445-67-7800	1000	1000	Administration	101	Corson, Bob
Bravo, Vicki	445-67-7800	1000	3000	Development	200	Doe, Suzanne
Corson, Bob	210-92-6518	2000	1000	Administration	101	Corson, Bob
Corson, Bob	210-92-6518	2000	3000	Development	200	Doe, Suzanne
Doe, Suzanne	253-44-7898	3000	1000	Administration	101	Corson, Bob
Doe, Suzanne	253-44-7898	3000	3000	Development	200	Doe, Suzanne

Note that restriction occurs through the WHERE statement, not through the SELECT statement. The SELECT statement is used for projection, as discussed above.

Joins

Joins provide the means of linking data in one table with data in another table and are frequently used in defining reports and queries.

A join is an operation that combines two tables to produce a third, subject to a restrictive condition. Every row of the new table must satisfy the restrictive condition.

Usually, when the two tables (A and B) are combined to form a third table (C), some condition is specified in the WHERE clause. This condition determines how a row from B is chosen to combine with a row from A. Often, this condition is equality, such that the value of a particular field from table A equals the value from a particular field from table B. Combining tables in this way is called an equijoin. Equijoins are often referred to as inner joins.

Although joins are often thought of as fundamental operators, they represent a combination of Cartesian Product plus Restriction.

Example For an example of an inner join, consider the following two tables, a “Suppliers” table and a “Parts” table:

Table 2-11: Suppliers Table

SNum	SName	SCity
S1	Smith	Paris
S2	Jones	London
S3	Blake	Boston
S4	Whitney	Boston
S5	Roberts	Paris

Table 2-12: Parts Table

PNum	PName	PCity
P1	Nut	Paris
P2	Screw	Houston
P3	Cog	New York
P4	Wheel	Boston
P5	Switch	Boston

Suppose you wish to query the database for the names of every supplier, part, and city, where the supplier and part are located in the same city. To do this, you would use the following SQL query:

```
SELECT SName,PName,SCity
FROM Suppliers,Parts
WHERE Suppliers.SCity=Parts.PCity
```

The SELECT clause specifies the fields to be included in the projected table. The Cartesian Product is derived from the tables named in the FROM clause of the above SQL statement, which combines the tables “Suppliers” and “Parts”. The WHERE clause in the above statement (Suppliers.SCity=Parts.PCity) specifies the inner join condition.

The Cartesian Product of the “Suppliers” table and “Parts” table is shown below. The rows that satisfy the inner join condition are shaded.

Table 2-13: Output Table After Cartesian Product Operation

SNum	SName	SCity	PNum	PName	PCity
S1	Smith	Paris	P1	Nut	Paris
S1	Smith	Paris	P2	Screw	Houston
S1	Smith	Paris	P3	Cog	New York
S1	Smith	Paris	P4	Wheel	Boston
S1	Smith	Paris	P5	Switch	Boston
S2	Jones	London	P1	Nut	Paris
S2	Jones	London	P2	Screw	Houston
S2	Jones	London	P3	Cog	New York
S2	Jones	London	P4	Wheel	Boston
S2	Jones	London	P5	Switch	Boston
S3	Blake	Boston	P1	Nut	Paris
S3	Blake	Boston	P2	Screw	Houston
S3	Blake	Boston	P3	Cog	New York
S3	Blake	Boston	P4	Wheel	Boston
S3	Blake	Boston	P5	Switch	Boston
S4	Whitney	Boston	P1	Nut	Paris
S4	Whitney	Boston	P2	Screw	Houston
S4	Whitney	Boston	P3	Cog	New York
S4	Whitney	Boston	P4	Wheel	Boston

Table 2-13: Output Table After Cartesian Product Operation (Continued)

SNum	SName	SCity	PNum	PName	PCity
S4	Whitney	Boston	P5	Switch	Boston
S5	Roberts	Paris	P1	Nut	Paris
S5	Roberts	Paris	P2	Screw	Houston
S5	Roberts	Paris	P3	Cog	New York
S5	Roberts	Paris	P4	Wheel	Boston
S5	Roberts	Paris	P5	Switch	Boston

This join operation yields the following output table:

Table 2-14: Output After Join Operation

SName	SCity	PName
Smith	Paris	Nut
Blake	Boston	Wheel
Blake	Boston	Switch
Whitney	Boston	Wheel
Whitney	Boston	Switch
Roberts	Paris	Nut

When the join is based on an exact match between fields from the two tables, it is a simple join. Rows in the both tables where no match is found do not appear in the output table. There are no rows in the output table for the cities, “London”, “New York”, or “Houston”.

InterSystems’ Extensions to the Relational Model

InterSystems has extended SQL and the relational model to include two additional types of joins: **one-way outer joins** and **implicit joins**.

One-Way Outer Joins

With standard “inner” joins, when rows of one table are linked with rows of a second table, a row in the first table that finds no corresponding row in the second table is excluded from the output table.

With one-way outer joins, all rows from the first table are included in the output table even if there is no match in the second table. The first table pulls relevant

information out of the second table but never sacrifices its own rows for lack of a match in the second table.

When specifying a one-way outer join, the order in which you name the tables in the FROM clause is very important. The first table you specify is the source table for the join.

You specify an outer join by using the symbol `=*` in place of `=` in the WHERE clause of the SQL query.

Example In the example below, the Suppliers table is specified as the source table for a one-way outer join operation (`=*`) with the Parts table, where the SCity field matches the PCity field.

```
SELECT SName,PName,SCity
FROM Suppliers,Parts
WHERE Suppliers.SCity=*Parts.PCity
```

The SELECT clause specifies the fields to be included in the projected table. The Cartesian Product is derived from the tables named in the FROM clause of the above SQL statement, which combines the Suppliers and Parts tables. The WHERE clause in the above statement (`Suppliers.SCity=*Parts.PCity`) specifies the one-way outer join condition.

This query returns all rows from the Suppliers source table as well as any rows from the Parts table where the SCity field matches the PCity field.

This join operation yields the following output table:

Table 2-15: Output After Join Operation

SName	SCity	PName
Smith	Paris	Nut
Jones	London	
Blake	Boston	Wheel
Blake	Boston	Switch
Whitney	Boston	Wheel
Whitney	Boston	Switch
Roberts	Paris	Nut

Implicit Joins

One of the most powerful features of the relational model is its ability to handle unanticipated ad hoc queries in a graceful and straightforward manner. However, for many implementations this capability is costly: the power to join tables in complex ways often means sacrificing the ease of executing joins for the more common connections between tables.

InterSystems has solved this problem by implementing the **implicit join**. Implicit joins are pre-defined joins between tables which you specify in the Data Dictionary. They allow you to define queries without specifying the WHERE condition that is used to join tables.

Open M with SQL supports two types of implicit joins, **designative references** and **characteristic relationships**.

Designative references and characteristic relationships are useful for:

- n Pre-defining commonly used joins
- n Improving data access efficiency
- n Formally specifying integrity constraints

Note You may only define designative and characteristic relationships among tables that reside within a single database.

Designative References

A designative reference is a many-to-one link between tables in which one field of the designating table contains the Row IDs of all rows in the designated table. A designative reference is said to be a non-dependent link because rows in the referenced table exist independently of rows in the designating table. In relational database terminology, the designating table has a “foreign key” on the referenced table. In M terminology, the designating table has a “pointer” to the referenced table. In Open M with SQL, a field that designates another table is called a designative reference field.

In the example below, the Customer field of the Invoice table serves as the Designative Reference field to rows in the Customer designated table:

Table 2-16: Invoice Table

InvNum	InvTotal	Customer
1234	100.00	C1
5555	20,000.00	C3
3333	5,000.00	C4

Table 2-17: Customer Table

Number	Name	Address
C1	Acme Hardware	10 Main Street Boston, MA
C2	Waterfront Motors	210 Willow Street Brighton, MA
C3	Global Furniture	1010 5th Street New York, NY
C4	Hill Pharmaceuticals	958 Jordan Ave. Pittsburgh, PA

The Designative Reference field, when it is not empty, contains a value that identifies one and only one row of the referenced table. Every entry in the Customer field of the Invoice table that is not empty must have exactly one corresponding entry in the Customer table. However, not all of the Number field values in the Customer table need appear in the Customer field of the Invoice table. In this way, a designative reference satisfies the relational definition of a one-way outer join.

Furthermore, when the “Invoice” table is linked by designative reference to the Customer table, the following is true:

- n There may be invoices with no “Customer” value, but
- n If a “Customer” value appears, there must also be a “Number” value in the “Customer” table with the same value, and
- n There may be customers with no invoices

Designative references contribute to data storage efficiency by helping to eliminate unnecessary redundancy. Since you can access the information through a designative reference to the Customer table, it is not necessary to store the customer names and addresses in the Invoice table. Accordingly, updates need be made in only one table, rather than in two or more tables.

A table may have several fields that designate the same or different tables. Similarly, a table may be designated by any number of tables. For example, the Invoice table may contain another designative reference field to the Accounts table. And the Accounts table might have a designative reference to the Customer table.

Characteristic Relationships

A characteristic relationship is a link between tables in which rows in one table (the “child table”) are existence-dependent on (cannot exist without) rows in another table (the “parent table”), such that parent rows have a one-to-many rela-

tionship with child rows. A child table always designates its parent table. For this reason, a characteristic relationship can be thought of as a kind of designative reference. However, a characteristic relationship is more restrictive than a designative reference since the join condition specifies that all rows of the child table must designate the same parent table row. In this way, a characteristic relationship satisfies the relational definition of an inner join.

Extending our previous example, the `Line_Items` table, shown below, is a child table of the `Invoice` table (its parent). The existence of the `Line_Items` table is entirely dependent on its parent `Invoice` table. If an invoice is deleted from the parent table, its line items become “orphaned”, and must also be deleted, or transferred to a “foster parent”. Within the Data Dictionary definition of a table, you may set a trigger that will function to automatically delete all child rows when its parent is deleted.

Each row of the `Invoice` table can have multiple line items, illustrating the one-to-many relationship:

Table 2-18: Invoice Table

InvNum	InvTotal	Customer
2222	10,000.00	C3
5555	20,000.00	C2
1234	100.00	C1

Table 2-19: Line_Items Table

InvNum	Item	UnitCost	Quantity	Amount
2222	Chair	200.00	10	2,000.00
2222	Desk	300.00	10	3,000.00
2222	Rug	250.00	4	1,000.00
2222	Bookshelf	200.00	30	6,000.00

An `Invoice` table row might also have no line items.

Though characteristic relationships may be perceived as hierarchical, they comply with the tenets of the relational model; just as other tables, child tables can be addressed through SQL without explicit reference to the parent table.

A parent table may have several child tables. For example, a `Patient` table may have `Visits`, `Medical_Problems`, and `Lab_Tests` as child tables. However, a child table may have only one parent. (The `Visits` table, for example, may be a child only of the `Patient` table.) A child table can never be “orphaned”, i.e., exist without a parent table.

Implicit Join Syntax

Implicit joins simplify the process of querying the database. By defining characteristic relationships and designative references in the Data Dictionary, you may take advantage of Open M with SQL's implicit join syntax to facilitate the definition of queries.

Arrow syntax, a dash followed by a greater-than symbol (->), is an InterSystems' SQL extension used to indicate an implicit join between tables. This syntax causes an additional outer join condition to be added implicitly to the WHERE clause and the joined table to be added implicitly to the FROM clause of an SQL query. See "Extended Arrow Syntax" on page 7-7 in Chapter 7, *Implicit Joins* .

In Open M with SQL, you may use arrow syntax in the following three cases:

1. To signify Designative References between tables
2. To signify Child-to-Parent References between tables
3. To signify Parent-to-Child References between tables

Implicit Join Syntax in a Designative Reference

If the field A.b designates table B, and x is a field in table B, the reference:

A.b->x

points to the value of x in the row of table B corresponding to A.b. It is interpreted as a reference to B.x with B added implicitly to the FROM clause and an additional outer join condition added implicitly to the WHERE clause.

For example, the following query retrieves the patient's name and patient's doctor's name for every patient who lives in Boston:

```
SELECT Patient.Pname, Patient.Doctor->Dname
FROM Patient
WHERE Patient.City = "Boston"
```

Assuming that Patient.Doctor is a designative reference to the Doctor table, the above query is equivalent to:

```
SELECT Patient.Pname, Doctor.Dname
FROM Patient, Doctor
WHERE Patient.City = "Boston"
      AND Patient.Doctor = *Doctor.Doctor
```

Implicit Join Syntax in a Child-to-Parent Reference

If P is the parent of C and x is a field in P, the implicit join syntax:

```
C.P->x
```

for a given row points to the value of x in that row's parent row. It is interpreted as a reference to P.x with P added implicitly to the FROM clause and an additional outer join condition added implicitly to the WHERE clause.

For example, given a parent table Customer with a child table Invoice, the following query:

```
SELECT Invoice.Customer->Name
FROM Invoice
WHERE Invoice.Number = 51140
```

is equivalent to:

```
SELECT Customer.Name
FROM Invoice, Customer
WHERE Invoice.Number = 51140
      AND Invoice.Customer = Customer.Customer
```

Implicit Join Syntax in a Parent-to-Child Reference

If P is the parent of C and x is a field in C, the implicit join reference:

```
P.C->x
```

for a given P row points to the value of x in a child row of that row. It is interpreted as a reference to C.x with C added implicitly to the FROM clause and an additional outer join condition added to the WHERE clause.

For example, given parent Customer with child Invoice, the following query for all invoices for all customers named Smith:

```
SELECT Customer.Invoice->Number
FROM Customer
WHERE Customer.Name = "Smith"
```

is equivalent to:

```
SELECT Invoice.Number
FROM Customer, Invoice
WHERE Customer.Name = "Smith"
      AND Customer.Customer = Invoice.Customer
```

Integrity Constraints

Implicit join definitions include built-in integrity constraints. For instance, the existence-dependent relationship between the Line_Items table and the "invoice table may be regarded as an integrity constraint: no line item can exist without a corresponding invoice.

The Open M with SQL relational Data Dictionary can be used to define other integrity constraints, such as:

- n Field validation code to enforce integrity constraints at the field processing level, such as affect field values, required fields, and field formats.
- n Triggers to enforce table integrity constraints, such as complex interactions between fields, or the prohibition of DELETES from the table.

Multi-Line Fields

Open M with SQL extends the relational database model by permitting the creation of multi-line fields. Such fields are useful for storing information about a single entity where that information spans several lines. A typical multi-line field might be used for an address or a block of comment text.

Open M with SQL treats the data in multi-line fields as a single entity, in accordance with First Normal Form principle of the relational model.

InterSystems' SQL supports the use of multi-line fields in input operations (INSERT and UPDATE statements) and output operations using INTO lists by creating an array and matching each line of the multi-line field to a node in the array. InterSystems' SQL also supports the naming of multi-line fields in the SELECT statement of SQL SELECT queries. It does not, however, allow the use of multi-line fields to perform comparisons or row ordering in an SQL SELECT query.

You can access pieces of multi-line field data using the M language, but this use of multi-line fields is not recommended, because it does not adhere to first normal form.

Using the Data Dictionary Interface

The interface to the Open M Developer environment, of which the Data Dictionary is a component, is itself an Open M with SQL application consisting of menus and forms.

This chapter provides a quick introduction to the Open M with SQL application development environment. It shows how to access the Data Dictionary and create a base table definition and also teaches you the basics of how to use menus and forms.

Specifically, this chapter covers the following topics:

- | | | |
|---|---|-----------|
| n | Accessing the Data Dictionary | page 3-2 |
| n | Understanding the Data Dictionary Interface | page 3-5 |
| n | Navigating in Forms | page 3-9 |
| n | Using On-Line Help | page 3-11 |
| n | Saving a Base Table Definition and Exiting the Form | page 3-13 |

Accessing the Data Dictionary

The Data Dictionary is the component of the Open M Developer environment in which you define the underlying structure of your database.

Procedure To access the Data Dictionary:

1. Change to a directory which contains an M database.

On ISM systems, the M database directory contains a OPENM.DAT file. If you want to create a new M database and are running on an ISM system, you can run the MSU utility to do so.

If you are using another version of M, consult your documentation.

2. Enter the Open M Developer environment.

The command you use depends on the M system you are running and your operating system. For example, if you are running on an ISM for UNIX system, you type the following command at the operating system prompt:

```
$mux
```

At the M programmer prompt, type the following command to enter Open M Developer:

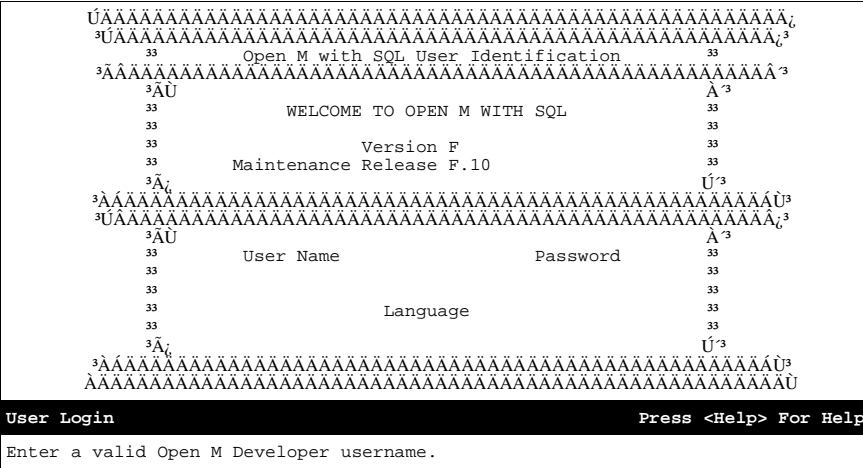
```
> do ^%msql
```

You see the Terminal Type: prompt, as shown below:

```
Terminal Type: VT220 =>
```

At the Terminal Type prompt, enter the terminal type you are currently using, or press <RETURN> to accept the default.

You see the Open M with SQL User Identification window, as shown below:



3. At the User Name field, enter your UserName, and press <RETURN>.

At the Password field, enter your password, and press <RETURN>.

At the Language field, enter the name of the language in which you want Open M with SQL to run.

Note: You may press the <SEARCH CURRENT TABLE> key to see a lookup box that lists all languages from which you may choose.

You see the Open M Developer Menu, as shown below:



4. From the Open M Developer Menu, select the Data Dictionary option.

To select an option from a menu, move the highlight bar on top of the option you want to select, and press <RETURN>. There are two ways to move the highlight bar:

- Use the <UP ARROW> and <DOWN ARROW> keys
- Type the initial characters of the option you want

Note: You may type *d* to select this option—it is a mnemonic accelerator.

You see the Data Dictionary menu, as shown below:

```
----- OPEN M Developer -----
+-----Data Dictionary-----+
|
| Base Table Definition
| View Definition
| Compile a Table
| Copy a Base Table Field
| Copy a View
| Change View's Starting Table
| Recreate all Default Structure Maps
| Populate Index Maps for a Table
| Reports on Data Dictionary
|
+-----+

Tuesday Mar 27, 1997                Directory: /us/morrow/
                                   Copyright (c) 1993 - InterSystems Corporation

Data Dictionary 09:55AM                Press <PF1><PF3> For Help
Press <RETURN> to select Base Table Definition.
```


Understanding the Data Dictionary Interface

The Data Dictionary interface is implemented as an Open M with SQL application, consisting of forms and windows.

A form consists of one master window and optionally one or more auxiliary windows. There are two types of forms in Open M with SQL:

- n Single-row forms
- n Multi-row forms

Single-Row Forms

Single-row forms display one row at a time. When you invoke a single-row form, you first see a row selection window (also called lookup window). At this row selection window, you can insert a new row or retrieve an existing one. After completing row selection, you see the form's master window, which solicits information for the newly inserted row or lets you edit information on the retrieved row.

Multi-Row Forms

Multi-row forms display multiple rows simultaneously. Multi-row forms have no row selection window. When you invoke a multi-row form, you first see the master window, which displays all (or a pre-defined subset of) existing rows. You can insert new rows or edit the information for existing rows.

Accessing the Base Table Definition Window

The Base Table Definition window is an example of a single-row form.

Procedure To access the Base Table Definition window:

1. From the Data Dictionary menu, select the Base Table Definition option.

```
UAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABase Table DefinitionAAAAAAAAAAAAAA;
3                                     3
3 Base Table Name                      3
3 _____                          3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
3                                  3
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AU
```

Base Table Definition	Selecting	Press <Help> For Help
-----------------------	-----------	-----------------------

- For each $\alpha \in \mathbb{N}$, let $\alpha!$ denote the factorial of α , that is, distinct from all

TABLE 1. "I" and "I'"

At the “Is this a new entry?” prompt, press **[RETURN]** to accept the “Yes”

- You see the Base Table Definition master window, as shown below:

Base Table Definition							
Base Table Name		Description					
gsm_test_____							
Use Default Physical Structure? Yes_							
< Fields >							
< M/WINDOWS Lookups >							
< Physical Structure >							
Approximate Number of Rows 250_____							
Implicit Joins	Validation Code	BaseTable Triggers	Compile Table	Comp Rel Objects	Copy Field	Advanced Options	Comments
Base Table Definition		Unsaved Data		Press <PF1><PF3> For Help			

Elements of a Window

An Open M with SQL window, such as the Base Table Definition window, has the following elements:

- n Data entry area
- n Status line
- n Message line (when one exists)
- n Horizontal options menu (when one exists)

Data Entry Area

An Open M with SQL window consists of as many lines as will fit on your terminal screen. One of those lines is reserved for the status line. Optionally, two more are reserved for the horizontal options menu and another for the message line. The remaining lines belong to the data entry area.

The data entry area is the main portion of the screen. It contains fields and their captions. It may also contain a window title and various other objects such as line drawing objects.

Required fields, where a value must be present in order to save the base table definition, appear in special typeface, depending on your terminal type. In this guide, they appear in **boldface**.

Status Line

The Status Line appears in reverse video between the data entry area and the horizontal options menu. It displays the following information:

- n Name of the current window
- n The appropriate keystrokes on your terminal for accessing the Open M with SQL Help Menu
- n Current form run-time mode

The table below lists and describes these form run-time modes:

Table 3-1: Open M with SQL Form Run-time Modes

Mode	Description
Selecting	Selecting mode occurs only in single-row forms. Also known as row selection, this mode is active whenever the user is selecting a row from the database.
Data Entry	In data entry mode, a user can edit or delete field values for existing rows or insert values for new rows. Any window that allows users to add, edit, or delete field values is a data entry window. In data entry mode, no message appears in the center of the status line until a field has been modified, at which time the message "Unsaved data" appears.
Unsaved Data	The form is in data entry mode and new data has been added/edited but not saved.
Inquiry	Inquiry mode prohibits users from modifying the information displayed on a window. In inquiry mode, users may navigate through the fields on a window to view their values, invoke branching fields and other triggers, back out of the current window, or proceed to the next window. However, users may not add, edit, or delete data.
/FILING...	This message flashes briefly in the middle of the status line whenever you save unsaved data.

Horizontal Options Menu

The horizontal options menu, if it exists, displays across the bottom of the screen, directly beneath the status line. This menu contains various options related to the current window.

Procedure To access an option on the Horizontal Options Menu:

1. From anywhere in the data entry area of a window, press the <GO TO BOTTOM MENU> key.

The cursor moves to the Horizontal Options Menu.

2. On the Horizontal Options Menu, you may select and invoke an option using any one of the following methods:
 - a. Use the arrow keys to position the cursor on the desired option, and press <RETURN> to invoke it.
 - b. Type the first letter of an option (the cursor selects and automatically invokes the option).
 - c. Press <CTRL-E> plus the first letter of the desired option.
 - d. If you are selecting an option from a submenu, you may type <CTRL-E>xy where *x* is the first letter of the primary menu option and *y* is the first letter of the submenu option.

Message Line

The **message line** is located just below the status line at the very bottom line of your display screen. The message line displays the following information:

- n Error message or Short Help message for the current field.

Note: Sometimes, fields are defined to display their Short Help messages automatically. In this case, you see the Short Help message as soon as the cursor lands on the field. If the field is not defined to display its Short Help message automatically, you see this message only when you press the <EXPLAIN> key.
- n Some system prompts, such as “Is this a new entry?”.

Navigating in Forms

This section describes how to navigate forms using keyboard actions.

Keyboard Actions

Since Open M with SQL supports many different terminals with different keyboards, InterSystems has developed its own terminology for keyboard actions. For instance, the keyboard action <GO TO BOTTOM MENU> moves the cursor down to the horizontal options menu and highlights the first menu option.

Open M with SQL maps each keyboard action to a particular key or keys for each terminal type. For example, on the DEC VT220 terminal type, you accomplish the <GO TO BOTTOM MENU> keyboard action by pressing the <F3> key. This guide and the software both reference the keyboard action names rather than actual key names on your terminal keyboard. The key names are listed in Appendix B, *Keyboard Actions*.

Learning the Keys Which Perform Actions on Your Keyboard

You may press the <HELP> key to see a list on-line that shows the keypress(es) assigned to each keyboard action on your terminal keyboard.

You may also reference Appendix B, *Keyboard Actions*, which contains a table that shows the key(s) assigned to each keyboard action for every terminal type. You may want to place a photocopy of the table for your terminal type near your keyboard.

Frequently Used Keyboard Actions

The following table lists and describes the keyboard actions you need to get started:

Table 3-2: Frequently Used Keyboard Actions

Keyboard Action	Description
<RETURN>	Selects an option from a lookup box or menu. Moves cursor to next field or activates a branching field in a window. Mimics <PROCEED> action when pressed when cursor is on the "<proceed>" prompt at the bottom of a window.
<RIGHT ARROW>	Moves cursor one character to the right in a field. Moves highlight bar to the next option in a menu.
<TAB>	Moves cursor to next field or menu option. Does not activate branching fields.
<GO TO BOTTOM MENU>	Moves cursor to the horizontal options menu at the bottom of the screen (if one exists) and highlights the first menu option.
<PROCEED>	Proceeds to the next logical step in a form. If you are on the form's master window, it saves unsaved data and exits the form. You can set a toggle that forces the PROCEED action to display a save menu before saving data and exiting a form. For more information on setting the PROCEED toggle, see the section of this chapter entitled "Saving a Base Table Definition and Exiting the Form" on page 3-13.
<PREVIOUS>	Closes the current window and moves back to the previous step in a form's development. If you are on the master window of a form, PREVIOUS exits the form — if there is unsaved data, it first displays a Save Menu to give you the option of saving before you exit. For more information on the Save Menu, see the section of this chapter entitled "Saving a Base Table Definition and Exiting the Form" on page 3-13.
<SAVE AND REMAIN>	Saves all unsaved data throughout the entire form and leaves the cursor in the same place where you invoked the key

Table 3-2: Frequently Used Keyboard Actions

<GETOUT>	Exits all the way out of a form, no matter where it is invoked. In a single-row form, GETOUT loops back to the row selection window. You can set a toggle that disables this looping behavior so that <GETOUT> exits the form back to the calling location. In a multi-row form, GETOUT always backs out to the calling location. If there is unsaved data, GETOUT first displays a Save Menu to give you the option of saving before you exit. For more information on the Save Menu, see the section of this chapter entitled “Saving a Base Table Definition and Exiting the Form” on page 3-13.
<GETOUTALL>	Exits the entire application and returns to the location where Open M with SQL was called. For example, if you called Open M with SQL from the M prompt, GETOUTALL returns you to the M prompt. Or, if you called Open M with SQL from some other program, GETOUTALL returns you to that program.

To learn more about using Form Generator forms, see the *Open M/SQL User Interface Programming Guide*.

Using On-Line Help

The Open M with SQL System Help Menu is a multiple screen on-line help facility that provides the following information:

- n Lists all Open M with SQL keyboard actions in subgroups of associated functions and shows the keystroke(s) you press to invoke them on your terminal type. For example, on the Save data and Exit menu you can see the corresponding keystrokes for all the exit keys— <PROCEED>, <PREVIOUS>, <GETOUT>, <GETOUTALL>, and <SAVE AND REMAIN>.
- n Lets you customize your run-time environment by activating or deactivating certain Open M with SQL options, such as:
 - PROCEED Save Menu toggle
 - Language setting
 - Insert/Typeover mode setting

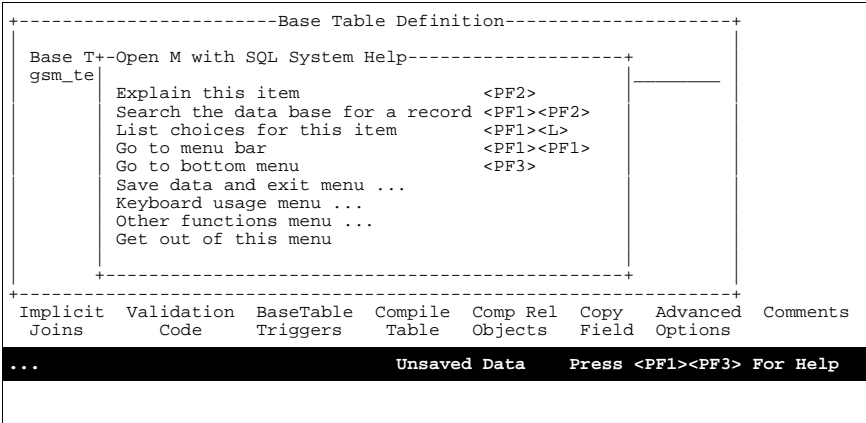
The preferences you define apply only to your local environment. Open M with SQL retains these settings across sessions. You define these run-time preferences in the Other Functions menu:

Procedure To access and use the Open M with SQL System Help menu:

- 1. From anywhere in the Open M Developer environment, press the <HELP> key.

Note: You can see the key(s) to press on your terminal type to invoke the <HELP> keyboard action on the far right side of the status line.

You see the Open M with SQL System Help menu, as shown below:



Menu items that are followed by a key or key sequence are keyboard actions. The key sequence represents the key(s) you press on your terminal to invoke the action. Alternatively, you can move the highlight bar to the menu item and press <RETURN> to invoke the corresponding action.

Menu items that are followed by an ellipsis (...) represent submenus that list more action keys. The actions are organized in functionally similar groups.

Only items shown in bold are active. Other items are not applicable to the location in the Open M with SQL development environment where you invoked the Help Menu.

- 2. Move your cursor to the menu item you want to activate, and press <RETURN>.

For example, when you select the Other Functions... menu item, you see the Other Function submenu, as shown below:

```

+-----Base Table Definition-----+
| Base T+-Other Functions-----+   |
| gsm_te |                         |   |
|         | Switch insert/typeover mode |   |
|         | Switch save menu show/don't show mode |   |
|         | User defined keys               <PF1><K> |   |
|         | Redraw Screen                   |   |
|         | Character Set                   |   |
|         | Language                       |   |
|         | Colors                         |   |
|         | Get out of this menu           |   |
+-----+-----+                     |
| Implicit Validation BaseTable Compile Comp Rel Copy Advanced Comments |
| Joins      Code      Triggers  Table  Objects  Field  Options           |
+-----+-----+-----+-----+-----+-----+-----+
| ...                               Unsaved Data  Press <PF1><PF3> For Help |
+-----+-----+-----+-----+-----+-----+

```

3. To move back to the previous screen in the Open M with SQL System Help menu, select the “Get out of this menu” option.

The “Get out of this menu” option emulates the <PREVIOUS> key.

4. To exit the Open M with SQL Help menu completely from any screen, press the <GETOUT> key.

Saving a Base Table Definition and Exiting the Form

When you create or edit a base table definition, you may use any of the following keyboard actions to save your unsaved data and exit definition form:

- n PROCEED
- n PREVIOUS
- n GETOUT
- n GETOUTALL

Typically, you should use the PROCEED action to save unsaved data and the PREVIOUS or GETOUT action to exit the form without saving unsaved data.

However, you can never leave a form that has unsaved data without explicitly choosing not to save the unsaved data. Saving occurs either automatically, or you see a menu that allows you to choose to save or not to save, as you prefer.

Setting the PROCEED Save Toggle

The default setting of the PROCEED Save toggle is established by the system manager in the System Configuration window.

Whatever the default, you may switch the mode for your own local environment by selecting the “Switch save menu show/don't show mode” option in the Other Functions submenu of the Open M with SQL System Help menu. Selecting this option automatically reverses the previous setting.

Procedure To change the setting of the PROCEED Save toggle:

1. Invoke a form in the Open M with SQL development environment and edit the data so there is unsaved data in the form.
2. From the master window of the form, press <PROCEED> key.

This tells you the current setting of the PROCEED Save toggle. If you see the Save on PROCEED menu, the toggle is ON. If you do not see the Save on PROCEED menu, the toggle is OFF.

3. Press the <HELP> key.

You see the Open M with SQL System Help menu.

4. Move the highlight bar to the Other Functions menu option, and press <RETURN>.

You see the Other Functions submenu.

5. Move the highlight bar to the “Switch save menu show/don't show mode” menu option, and press <RETURN>.

This changes the toggle setting.

6. Press the <GETOUT> key to exit the Open M with SQL System Help menu.

Designing a Sample Application

This chapter provides a brief overview of the process for designing a relational database application and introduces you to the sample application used throughout this guide. The sample application is a Documentation Tracking system.

Specifically, this chapter covers the following topics:

- | | | |
|---|--|-----------|
| n | Designing a Relational Database | page 4-2 |
| n | Overview of Sample Application | page 4-4 |
| n | Base Tables in Documentation Tracking System | page 4-5 |
| n | Reports in Documentation Tracking System | page 4-15 |

Designing a Relational Database

The following five steps outline the process for developing a complete relational database application in Open M with SQL:

1. Map the functional specifications for the application out on paper.
2. Define the data structures in the Data Dictionary, along with integrity constraints, table relationships, and triggers.
3. Use the Form Generator to develop data entry and inquiry forms.
4. Use M/PACT to develop reports.
5. Use the Menu Generator to tie the different parts of the application together.

Mapping the Functional Specifications

The first step in developing a relational database application is to organize the logical structure of the data on paper as a series of tables, just as you might design a paper filing system. To do this, consider the functional needs of your application and make a list of all the different pieces of data that you need. The object of the design is to describe the tables that constitute the database and how those tables will interact with one another. For example, one application might have a customer table, an invoice table, an invoice line item table, a parts table, etc. The customer table might then consist of fields for customer name, address, phone, current balance, etc.

You should also consider the data you need at the field level. For each field, identify its characteristics: name, data type, etc. It may be useful to develop a naming convention for fields if base tables share similar field names. This makes it easy to identify the base table to which a field belongs when you are working with views. For example, if you have a base table “Patients”, you might define all of its fields to use the suffix “_Pat”. Using a suffix rather than a prefix avoids having the same pattern at the beginning of all fields, thus allowing lookups to work with fewer keystrokes.

You may also want to design on paper the forms and reports your application will use. This can help ensure that you have included all the necessary data and properly defined the relationships between base tables.

Defining the Data Structure in the Data Dictionary

When you complete your design on paper, you are ready to define base tables and views in the Data Dictionary.

In the Data Dictionary, you will define integrity constraints on your data. Integrity constraints may be very simple; for example, you can specify a range of valid numbers that can be entered for a field. Other constraints may be more complex; perhaps you want to specify that an invoice cannot be entered into the invoice table without a corresponding customer entry in the customer table. Even more complex constraints may require the addition of complete SQL queries or procedural M code.

You will also define the relationships between base tables. For example, where rows of an invoice line item table cannot exist without a corresponding row in an invoice table, the line item table should be defined as a **characteristic table** of the invoice table. One field in the invoice table should also be defined to serve as a **designative reference** to rows in the customer table. Open M with SQL uses this information to enforce integrity constraints and produce automatic joins in queries.

You may also define processing triggers in the Data Dictionary. Triggers are sequences of actions defined to automatically occur, or be *triggered*, when certain other events occur. A trigger definition usually consists of an SQL query or M code segment that is invoked when rows are created, modified, or deleted. For example, if a medical record for a given patient is deleted, a trigger may automatically delete all of that patient's lab test information from various lab files. For further details see Chapter 11, *Base Table Triggers* .

See Chapter 1, *Introduction to the Data Dictionary* and Chapter 2, *The Open M with SQL Relational Database* , to learn about design concepts that save disk space, maintain database integrity and enhance programmer productivity.

Overview of Sample Application

The remainder of this chapter describes the design of the sample application used throughout Part II: *Basics* of this guide. The description includes many caveats that may help you as you design your own relational database applications.

The sample application is the InterSystems Documentation Tracking system. Its purpose is to track editions of all InterSystems guides as well as orders for guide covers and edition copies.

This sample application consists of seven base tables, one view, and several forms and reports. This guide is primarily concerned with the base tables and view. However, we make occasional reference to the forms and reports for illustration purposes.

The sample base tables contain fields that cover every data type supported by Open M with SQL. In addition, these tables demonstrate the use of many advanced features of the Data Dictionary environment, including lookups, triggers, and base table validation code.

Sample Application Uses Default Physical Structure

All the base tables in the Documentation Tracking system use default physical structure. As a result, the Data Dictionary automatically creates index maps for all fields defined as lookup fields in these base tables.

If you define a customized physical structure for a base table, the Data Dictionary does not automatically create index maps on any fields—you must define them manually.

Base Tables in Documentation Tracking System

The table below summarizes the base tables included in the Documentation Tracking system:

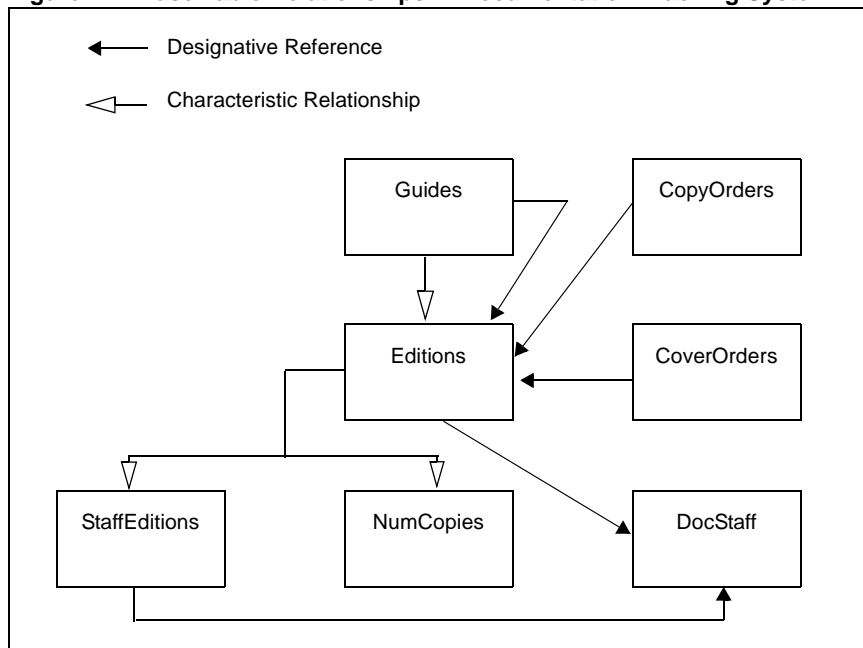
Table 4-1: Base Tables in Documentation Tracking System

Table Name	Description
DocStaff	Contains names of and information about InterSystems technical writers.
Guides	Contains names of and information about each InterSystems guide.
Editions	Child of Guides. Contains version and revision date of each edition of each Guide, as well as other information, such as new features, main author, and number of pages.
StaffEditions	Child of Editions. Contains names of other technical writers who worked on an edition and their role.
NumCopies	Child of Editions. Contains number of copies of an edition ordered by year.
CopyOrders	Contains information about each order placed for an edition, such as number of copies, vendor and cost.
CoverOrders	Contains information about each order placed for an edition, such as number of covers, vendor and cost.

Relationships Between Base Tables

The diagram below describes the relationships between the base tables in the Documentation Tracking system.

Figure 4-1: Base Table Relationships in Documentation Tracking System



Base Table Fields

This section summarizes the fields in each base table in the Documentation Tracking system. The first field in the table is the RowID field.

DocStaff Base Table

The table below lists and describes the fields in the “DocStaff” base table:

Table 4-2: Fields in DocStaff Base Table

Field	Data Type	Required?	Unique?	Lookup Field?
DocStaff	RowID	Yes	Yes	No
Name	Name	Yes	Yes	Yes
Phone	Text	No	No	No
Street	Text Multi-line	No	No	No
City	Text	No	No	No
State	Text Max. Length — 2	No	No	No
Country	Text	No	No	No
Zip	Number Max. Length — 9 Checks for 5 or 9 digits. If 9, validation code adds a dash character (-).	No	No	No
HireDate	Date	No	No	No
ServiceLength	Number Decimal places — 2 Computed field (result always equal to the computation)	No	No	No

Design Notes for DocStaff Base Table

This base table contains information about each technical writer in the InterSystems Documentation department.

The “Phone” field is defined as a text field with no validation code, which allows for the input of international phone numbers.

The address information is geared to the United States. It is broken up into separate fields, since there is sometimes a need to sort by city, state, or zip code. The “Street” field is a multi-line field to allow for both a street address and a suite or apartment number.

The “State” field permits two alphabetic characters, as used in standard postal abbreviations. A field validation routine checks for a legal value by comparing input to a list of the standard state abbreviations. Another way to do this is to create a separate base table that contains one row for each state abbreviation and then make the “State” field a designative reference to that table. If you elected to customize the physical structure for this state table, you could define the postal abbreviation as the RowID field, which is possible since each postal abbreviation is unique. This would improve the processing efficiency of your application.

The “ServiceLength” field is a computed field based on the value of another field, “HireDate”. For more information on defining computed fields, see Chapter 6, *Defining Base Table Fields*.

Guides Base Table

The table below lists and describes the fields in the “Guides” base table:

Table 4-3: Fields in Guides Base Table

Field	Data Type	Required?	Unique?	Lookup Field?	Computed?
Guides	RowID	Yes	Yes	No	No
Title	Text	Yes	Yes	Yes	No
Product	Multiple Choice: n Open M n RDBMS n Server n M/NET	No	No	No	No
LatestEditionGuide	Designative Reference (to Editions table)	No	No	No	Yes — by a trigger in Editions when a new edition is inserted
NumEditions	Number	No	No	No	Yes — by a trigger in Editions
Retired	Yes/No Default is No	Yes	No	Yes	No

Table 4-3: Fields in Guides Base Table (Continued)

Field	Data Type	Required?	Unique?	Lookup Field?	Computed?
DateRetired	Date	Maybe If {Retired }=Y	No	No	No
WhoRetired	Name	Maybe If {Retired }=Y	No	No	No

Design Notes for Guides Base Table

The “Guides” base table contains entries for all InterSystems guides. Since there are multiple editions of each guide, we create a child table of the “Guides” table called “Editions”. For each guide in the “Guides” parent table, there can be many editions in the “Editions” child table, each of which represents one edition of the corresponding parent guide.

The “Guides” table contains a Designative Reference field called “LatestEditionGuide”, which points to the “Editions” child table. This field is defined to have an Output-Only protection level because its value is computed automatically via a base table trigger of type Routine located in the “Editions” table. When a new row is inserted in the “Editions” table, a post-filing INSERT trigger updates the “LatestEditionGuide” field in the “Guides” table, if necessary. For more information about defining such a trigger, see Chapter 11, *Base Table Triggers*, where one of the examples describes this trigger and the M routine it calls.

The “Guides” table also contains a field called “NumEditions”. Like “LatestEditionGuide”, this field is defined to have an Output -Only protection level and is computed by post-filing INSERT and DELETE triggers in the “Editions” table. When a new edition is inserted, a trigger increments this field by 1. If an existing edition is retired, a trigger decrements this field by 1.

Note also that the field “Retired” is a required Yes/No field with a default value of “No”. For more information about Yes/No fields, see “Defining a Yes/No Field” on page 6-27 in Chapter 6, *Defining Base Table Fields*.

The fields “DateRequired” and “WhoRetired” are both conditionally-required fields. For more information about conditionally-required fields, see “Defining a Required Field” on page 6-49 in Chapter 6, *Defining Base Table Fields*.

Editions Base Table

The table below lists and describes the fields in the “Editions” base table:

Table 4-4: Fields in Editions Base Table

Field	Data Type	Required?	Unique?	Lookup Field?
Editions	RowID (This field is computed based on the Guides and childsub fields.)	Yes	Yes	No
Guides	Parent Reference — RowID of corresponding row in parent table.	Yes	No	No
childsub	RowID of corresponding row in parent table.	Yes	No	No
Version	Text	Yes	No	Yes
RevisionDate	Date	Yes	No	No
NumPages	Number	No	No	No
Price	Number	No	No	No
MainAuthor	Designative Reference (to DocStaff table)	Yes	No	Yes
AuthorRole	Text Multi-line	No	No	No
Summary	Features added in this edition	No	No	No
Retired_Ed	Yes/No Default is No	Yes	No	No
DateRetired_Ed	Date	Maybe If {Retired}= Y	No	No
WhoRetired_Ed	Name	Maybe If {Retired}= Y	No	No

Design Notes for Editions Base Table

Editions of InterSystems documentation are currently identified by the version of the product for which they were produced and the date the edition was printed. This information appears on the title page of each guide. The version of a guide is stored in the “Version” field, and the print date is stored in the “RevisionDate” field.

Although a title might still be active, a particular edition of that title may be retired. Thus, we include the three retired fields in the “Editions” base table with a special “_Ed” suffix to distinguish these fields from their analogues in the “Guides” base table.

StaffEditions Base Table

The table below lists and describes the fields in the “StaffEditions” base table:

Table 4-5: Fields in StaffEditions Base Table

Field	Data Type	Required?	Unique?	Lookup Field?
StaffEd	RowID This is a computed field based on the Editions and childsub fields.	Yes	Yes	No
Editions	Parent Reference — RowID of corresponding row in parent table. (This is one of the fields on which the RowID is based).	Yes	No	No
childsub	RowID of row in this table. (This is one of the fields on which the RowID is based).	Yes	No	No
NameEdstaff	Designative Reference (to DocStaff table)	Yes	No	No
Role	Text Multi-line	No	No	No

Design Notes for StaffEditions Base Table

Since we cannot know how many writers might be assigned to work on a particular edition of a guide, it makes more sense to store these additional writers in a child table rather than provide a limited number of fields.

NumCopies Base Table

The table below lists and describes the fields in the “NumCopies” base table:

Table 4-6: Fields in NumCopies Base Table

Field	Data Type	Required?	Unique?	Lookup Field?
NumCopy_Ed	RowID This is a computed field based on the Editions and childsub fields.	Yes	Yes	No
Editions	Parent Reference — RowID of corresponding row in parent table. (This is one of the fields on which the RowID is based).	Yes	No	No
childsub	RowID of row in this table. (This is one of the fields on which the RowID is based).	Yes	No	No
Year	Number	Yes	No	No
NumCopies	Number	No	No	No

Design Notes for NumCopies Base Table

The “NumCopies” is a child table of “Editions”. This table tracks the number of copies printed for each edition of each InterSystems guide per fiscal year.

When a row is inserted in the “CopyOrders” table, a base table post-filing SQL trigger updates the “NumCopies” field in the “NumCopies” table for the current year.

CopyOrders Base Table

The table below lists and describes the fields in the “CopyOrders” base table:

Table 4-7: Fields in CopyOrders Base table

Field	Data Type	Required?	Unique?	Lookup Field?
CopyOrders	RowID This is a computed field based on the Editions and childsub fields.	Yes	Yes	No
Editions	Parent Reference — RowID of corresponding row in parent table. (This is one of the fields on which the RowID is based).	Yes	No	No
childsub	RowID of row in this table. (This is one of the fields on which the RowID is based).	Yes	No	No
EditionCopy	Designative Reference (to Editions table)	Yes	No	No
Vendor	Text	Yes	No	Yes
NumCopies	Number	Yes	No	No
OrderDate	Date	Yes	No	No
ETA	Date	No	No	No
ArrivalDate	Date	No	No	No
Cost	Number	No	No	No
PONum	Text	No	No	Yes

Design Notes for CopyOrders Base Table

The “CopyOrders” base table tracks the print orders for each edition of each InterSystems guide.

Note that the “CopyOrders” table has a base table post-filing INSERT trigger that updates the “NumCopies” field in the “NumCopies” table.

CoverOrders Base Table

The table below lists and describes the fields in the “CoverOrders” base table:

Table 4-8: Fields in CoverOrders Base Table

Field	Data Type	Required?	Unique?	Lookup Field?
CoverOrders	RowID	Yes	Yes	No
EditionCover	Designative Reference (to Editions table)	Yes	No	Yes
Master?	Yes/No	Yes	No	Yes
Vendor	Text	Yes	No	Yes
OrderDate	Date	Yes	No	No
ETA	Date	No	No	No
ArrivalDate	Date	No	No	No
NumCovers	Number	Yes	No	No
Cost	Number	No	No	No

Design Notes for CoverOrders Base Table

The “CoverOrders” base table tracks the print orders for book covers associated with each edition of each InterSystems guide.

Reports in Documentation Tracking System

The Documentation Tracking system includes the following reports:

- n Summary information about each edition of each guide
- n Summary of all guides for which a particular staff member was the main author
- n Orders of guides and guide covers on a per edition basis



Basic Operations

Chapter 5

Defining a Base Table

Chapter 6

Defining Base Table Fields

Chapter 7

Implicit Joins

Chapter 8

Base Table Lookups

Chapter 9

Index Maps

Chapter 10

Data Conversion and Validation

Chapter 11

Base Table Triggers

Chapter 12

Base Table Help and Error Messages

Chapter 13

Field Help and Error Messages

Chapter 14

Views

Chapter 15

Data Dictionary Reports

Defining a Base Table

This chapter describes how to create, define, edit, and delete a base table in the Data Dictionary.

Specifically, it covers the following topics:

n	Base Table Definition Overview	page 5-2
n	Defining a Base Table	page 5-4
n	Defining a Child Table	page 5-11
n	Generating Default Physical Structure	page 5-15
n	Compiling a Base Table	page 5-17
n	Compiling Related Objects	page 5-20
n	Editing a Base Table Definition	page 5-21
n	Deleting a Base Table Definition	page 5-23

Base Table Definition Overview

This chapter presents a checklist for defining a base table and provides detailed information about how to perform several steps in the checklist.

We recommend that you use the information in this chapter as follows:

1. Make copies of the Base Table Definition Checklist. Use this list to check off the steps as you define each base table in your application.
2. For each step in the checklist, use the associated chapter reference to find more information.

In the designated chapter for each step you will find both conceptual information and a step-by-step procedure for completing the step. Read the conceptual information first and then follow the procedure to complete the step.

The procedures for many of the steps use examples from the Documentation Tracking sample application described in Chapter 4, *Designing a Sample Application*.

Base Table Definition Checklist

The checklist below lists the top-level steps for defining a base table.

Table 5-1: Base Table Definition Checklist

	#	Required?	Action	Where to Find More Information
r	1	Yes	Create a base table.	<i>"Defining a Base Table" on page 5-4.</i>
r	2	Yes	Specify default or customized physical structure.	<i>"Generating Default Physical Structure" on page 5-15.</i> Chapter 16, <i>Default Physical Structure</i> <i>"If You Started Using Customized Physical Structure" on page 18-6 in Chapter 18, Creating a Customized Map Definition .</i>
r	3	Recommend	Define characteristic relationships.	<i>"Defining a Child Table" on page 5-11.</i> <i>Chapter 7, Implicit Joins .</i>
r	4	Yes	Define base table fields.	<i>Chapter 6, Defining Base Table Fields .</i>
r	5	Recommend	Define form row selection lookups.	<i>Chapter 8, Base Table Lookups .</i>

Table 5-1: Base Table Definition Checklist (Continued)

r	6	Optional	Perform mapping.	Chapter 16, <i>Default Physical Structure</i> . Chapter 18, <i>Creating a Customized Map Definition</i> .
r	7	Recommend	Define indexes.	Chapter 9, <i>Index Maps</i> (for tables using default physical structure). Chapter 18, <i>Creating a Customized Map Definition</i> (for tables using customized physical structure)
r	8	Required (default supplied)	Specify the approximate number of rows the table will contain.	" <i>Defining a Base Table</i> " on page 5-4.
r	9	Optional	Define validation constraints to be met before a row of base table can be saved.	Chapter 10, <i>Data Conversion and Validation</i> .
r	10	Optional	Define base table triggers.	Chapter 11, <i>Base Table Triggers</i> .
r	11	Optional	Define base table routine name prefix.	" <i>Base Table Routine Names</i> " on page 5-17.
r	12	Yes	Save the base table definition.	Chapter 5: <i>Creating a Base Table</i> .
r	13	Yes (if specified in step 2)	Generate default physical structure	" <i>Generating Default Physical Structure</i> " on page 5-15.
r	14	Yes	Compile the table.	" <i>Compiling a Base Table</i> " on page 5-17.
r	15	Optional	Compile base table's related objects	" <i>Compiling Related Objects</i> " on page 5-20.

Note If you plan to perform customized mapping for the base tables you create (rather than use the default physical structure), read Chapter 17, *Relational Definition of an M Database* to understand the order in which you should perform the steps in this checklist.

Defining a Base Table

Step 1 in the Base Table Definition Checklist is to create (define) a new base table.

You may create as many base tables in your database as you wish.

Procedure To create a base table definition:

1. From the Data Dictionary menu, select the Base Table Definition option.

You see the Base Table Definition window, as shown below:

-----Base Table Definition-----

Base Table Name

Base Table Definition

Selecting

Press <PF1><PF3> For Help

2. At the Base Table Name field, enter a unique base table name, then press <RETURN>.

A base table name may range from 1 to 40 characters in length. It may contain alphanumeric characters (letters and numbers) and the “_” (underscore) character. It may not contain blank spaces or other punctuation characters. A base table name must begin with a letter. Do not use a SQL reserved word as a base table name; SQL reserved words are listed in Appendix C, *SQL Reserved Words*.

To create a new base table, you must enter a unique name, distinct from all existing base table names in the current database. The base table name is displayed exactly as you typed it; however, the test of whether it is unique is performed on its internal representation. Internally, base table names are case-insensitive. Therefore, for example, “ACCOUNTS” and “Accounts” are considered to be the same name. The underscore character is removed from the internal representation. Therefore, “Unitspending”, “Unit_spending” and “Units_pending” would all be considered to be the same name.

Use care in selecting base table names. The purpose of the base table should be clearly indicated by the base table name and the base table description. The relationship between the base table and other base tables should be clearly indicated by the base table name and the base table description. You can later change the name of a base table, but its initial name continues to be used as the Row ID name and as an element in the Master Map. For this reason, any change to a base table name should be commented. See “Developer Comments” on page 12-11 in Chapter 12, *Base Table Help and Error Messages*.

You can use the initial characters of the base table name to select a Series of Base Tables report, as described “Series of Base Tables Report” on page 15-7 in Chapter 15, *Data Dictionary Reports*. For this reason, you may want to specify the same initial letter(s) for the names of related base tables.

If a base table with the name you specified already exists, a popup box appears with the name and description of that base table.

3. The “Is this a new entry?” prompt appears just below the status line. At this prompt, press <RETURN> to accept the “Yes” default and confirm that you are creating a new base table. Specifying “No” and pressing <RETURN> returns you to the Base Table Name field.

If the base table name is invalid, a message appears at this point.

4. You see the Base Table Definition main window, as shown below:

Base Table Definition							
Base Table Name				Description			
Guides_____				Titles of InterSystems Guides_____			
Use Default Physical Structure? Yes_							
< Fields >							
< M/WINDOWS Lookups >							
< Physical Structure >							
Approximate Number of Rows 250_____							

Base Table Definition		Unsaved Data			Press <PF1><PF3> For Help		
Implicit Joins	Validation Code	BaseTable Triggers	Compile Table	Comp Rel Objects	Copy Field	Advanced Options	Comments

The table below lists and describes the fields located on the Base Table Definition window. Press the <RETURN> key after entering a value for each field:

Table 5-2: Fields on Base Table Definition window

Field Name	Description
Base Table Name	This is a required field, which you entered on the previous screen. You can change this base table name at any time to another unique name; you do not have to recompile the base table. If you change the name of an existing base table, Open M with SQL automatically updates all references to the base table to reflect the new name. However, values taken from the base table name, such as the name of the Row ID field and the TableName portion of the master map (new format) do not change, but retain the initial name of the base table; this has no effect on base table functionality. It is recommended that you comment all changes to the base table name (see “Developer Comments” on page 12-11 in Chapter 12, <i>Base Table Help and Error Messages</i>). If you change the base table name, you may need to edit queries to refer to this new name.
Description	This is an optional field. Enter a brief description of the base table. The description may be up to 35 characters in length. You may use any characters, including blank spaces and punctuation symbols. This description appears with the base table name in listings of existing base tables, and as the column header (unless overridden) for M/PACT reports. To record more detailed comments for the base table, see “Developer Comments” on page 12-11 in Chapter 12, <i>Base Table Help and Error Messages</i> .

Table 5-2: Fields on Base Table Definition window (Continued)

Field Name	Description
Use Default Physical Structure?	<p>This is a required field. Specify <i>Yes</i> or <i>No</i> to indicate whether or not you want the Data Dictionary to create a default physical structure for your base table. The physical structure of a base table is the mapping of its fields to an underlying M global structure.</p> <p>If you answer <i>Yes</i>, The Data Dictionary will create a default physical structure for your base table. <i>Yes</i> is the default response. If you are creating a new database, and especially if you don't have much experience with M, you should use the default physical structure. For more information about the default physical structure, see Chapter 16, <i>Default Physical Structure</i> .</p> <p>If you answer <i>No</i>, the Data Dictionary will not create a default physical structure, and you must custom-define the underlying physical structure for your base table. If you are creating this table to provide a relational view of an existing M database, you must define a customized physical structure. To learn how to define a customized physical structure, see Chapter 18, <i>Creating a Customized Map Definition</i> .</p> <p>Warning: If you answer <i>No</i> to the Default Physical Structure? field and then save the base table, you can never reenabte the default physical structure. From that point on for the life of the base table you will be responsible for mapping any modification you make to the logical structure.</p>
<Fields>	<p>Press <RETURN> to access the Field Definition window, where you may define fields for your base table. You may define fields at this point, or later after creating and saving the base table. Creating a table automatically defines one field within that table, known as the Row ID field. You may later modify the default definition of this field.</p> <p>For more information on creating and defining base table fields, see Chapter 6, <i>Defining Base Table Fields</i> .</p>
<M/WINDOWS Lookups>	<p>Optionally, press <RETURN> to access the Lookup Specification window where you may define default row selection behavior to be used by all forms that are associated with the current base table.</p> <p>For more information on defining row selection behavior, see Chapter 8, <i>Base Table Lookups</i> .</p>

Table 5-2: Fields on Base Table Definition window (Continued)

Field Name	Description
<Physical Structure>	<p>Optionally, press <RETURN> to access the Default Physical Structure window where you may do any of the following:</p> <ul style="list-style-type: none"> n Change the default global name n Examine the default physical structure <p>For more information about the default physical structure, see Chapter 16, <i>Default Physical Structure</i> .</p> <ul style="list-style-type: none"> n Define a customized physical structure, including index maps <p>For more information about defining a customized physical structure, see Chapter 18, <i>Creating a Customized Map Definition</i> .</p>
Approximate Number of Rows	<p>This is a required field for which a default is provided. If you delete the default value, you must replace it with another numeric value.</p> <p>Enter an approximation of the number of rows you expect this table to contain. The Query Optimizer uses this value to decide how to most efficiently access the rows in the table when processing a query or report. The default is 250.</p> <p>The figure you enter here need not be precise—you can use a “ballpark figure” such as 20, 200, or 200000. Enter the number without commas or other punctuation.</p> <p>For a child table, enter the total number of rows in the table, calculated as: {# of parent rows} x {# of child rows for each parent row}.</p> <p>If the number of rows in a table changes significantly (for example, due to expansion of the database), update this value to represent the current number of rows.</p>

5. When you finish defining your base table, press the <PROCEED> key to save your definition and exit the Base Table Definition window.

Horizontal Options Menu

Located at the bottom of the Base Table Definition window is the Base Table Definition horizontal options menu. These options provide additional functionality to the base table definition environment.

To access the Base Table Definition horizontal options menu, press the <GO TO BOTTOM MENU> key from within the Base Table Definition main window.

To select an option from the Base Table Definition horizontal options menu, use the <RIGHT ARROW> and <LEFT ARROW> keys to position the select bar on a desired option, then press <RETURN>. Alternatively, you can select an option by typing the initial letter of the desired option.

The table below lists and describes the eight options on the Base Table Definition horizontal options menu:

Table 5-3: Base Table Definition Horizontal Options Menu

Option	Meaning
Implicit Joins	<p>Select this option to access a Joins window, where you may define new characteristic relationships (child tables) for the current base table or view all existing characteristic relationships and designative references defined for the current base table.</p> <p>For more information about defining characteristic relationships among tables, see “Defining a Child Table” on page 5-11, and Chapter 7, <i>Implicit Joins</i> .</p> <p>For more information about defining designative references to other tables, see “Defining a Designative Reference Field” on page 6-14 in Chapter 6, <i>Defining Base Table Fields</i> .</p>
Validation Code	<p>Select this option to access the Base Table Validation Code window, where you may supply validation code to be tested before an INSERT, UPDATE, or DELETE operation is performed on the base table.</p> <p>For more information about defining base table validation code, see “Base Table Validation Code” on page 10-15 in Chapter 10, <i>Data Conversion and Validation</i> .</p>
BaseTable Triggers	<p>Select this option to access the Base Table Triggers window, where you may define Pre-Filing and/or Post-Filing triggers to be associated with an INSERT, UPDATE, or DELETE operation on the base table.</p> <p>For more information about defining base table triggers, see Chapter 11, <i>Base Table Triggers</i> .</p>
Compile Table	<p>Select this option to compile the current base table. See “Compiling a Base Table” on page 5-17.</p>
Comp Rel Objects	<p>Select this option to use the Compile Related Objects utility, which creates a compilation configuration that includes the current base table and all objects related to the current base table. You can modify the configuration list and execute it to sequentially compile all objects on the list.</p> <p>For more information about using the Compile Related Objects utility, see “Compiling Related Objects” on page 5-20.</p>
Copy Field	<p>Select this option to access the Field Copy utility, which lets you copy definitions of base table fields within a base table or between base tables in the Data Dictionary.</p> <p>For information about using the Field Copy utility, see “Copying a Field Definition” on page 6-57 in Chapter 6, <i>Defining Base Table Fields</i> .</p>

Table 5-3: Base Table Definition Horizontal Options Menu (Continued)

Option	Meaning
Advanced Options	<p>Select this option to access the Base Table Advanced Options window, where you may do all of the following:</p> <ul style="list-style-type: none"> ▮ Determine who is the owner of the base table (read-only). ▮ Specify a routine name to be used for compiling the current base table (see “Base Table Routine Names” on page 5-17). ▮ Associate help topic definitions with the base table (see “Help Text” on page 12-5 in Chapter 12, <i>Base Table Help and Error Messages</i> . ▮ Define customized messages for row insert and delete (see “Row Insert and Delete Messages” on page 12-7 in Chapter 12, <i>Base Table Help and Error Messages</i> . ▮ Specify an Override Lock Reference. This field allows you to specify a locking reference expression other than the full global reference of the master map. See “Examining a Default Master Map” on page 16-12 in Chapter 16, <i>Default Physical Structure</i> . This feature is used for locking/unlocking rows both in forms and in SQL queries. ▮ Enable Field Length Check ▮ Table ID# (irn) displays the internal reference number of the current base table. This is a read-only field. The irn is used in the old format Master Map (see “New and Old Master Map Structures” on page 16-12 in Chapter 16, <i>Default Physical Structure</i> .
Comments	<p>Select this option to record developer’s notes and comments on the base table. See “Developer Comments” on page 12-11 in Chapter 12, <i>Base Table Help and Error Messages</i> .</p>

Defining a Child Table

A characteristic relationship is a link between base tables in which rows in one table (a “child table”) are existence-dependent on (cannot exist without) rows in another table (a “parent table”), such that parent rows have a one-to-many relationship with child rows. A row in the parent table can point to many rows in the child table, but each row in the child table points to only one row in the parent table.

When you define a characteristic relationship, you are creating a child table.

For a complete discussion of characteristic relationships, see Chapter 7, *Implicit Joins*.

Creating a Child Table from the Parent Table

You create a child table by defining it in the base table definition of its parent. When you do this, the Data Dictionary automatically generates a skeletal default structure for the child table, including its Row ID field, and inserts an entry into the list of existing base tables. For more information about the default structure of a child table, see “Row ID Definition in Child Tables” on page 16-5 in Chapter 16, *Default Physical Structure*.

You may complete the definition of your child table by selecting it from the list of existing base tables and then editing the definition to meet your requirements.

Warning: Once you have created a child table, it must always remain a child table. If you decide this table should really be an independent table, you must recreate the table from scratch.

Procedure To define a child table

1. Open the Base Table Definition window of the base table for which you want to define a child table. That is, open what will be the parent table.
2. Press the <GO TO BOTTOM MENU> key to access the Base Table Definition horizontal options menu.
3. From the Base Table Definition horizontal options menu, select the Implicit Joins option.

You see the Characteristic/Designative Joins window, as shown below:

-----Base Table Definition-----

Base Table Name	Description
Guides	

-----Characteristic Joins-----

Parent Table	Child Tables (1/0)

-----Designative Joins-----

References Tables (0/0)	From Field	Description
-------------------------	------------	-------------

...Characteristic/Designative JoinsUnsaved Data Press <PF1><PF3> For Help

Here we are only concerned with the Characteristic Joins portion of this window.

- 4. At the Child Tables multi-line field, enter the name of the child table you want to create, and press the <PROCEED> key.

The child table name follows all of the naming conventions of a base table name, as described earlier in this chapter.

You may create as many child tables for a particular parent as you wish, but you must create them one at a time. Completing the definition of a child table returns you to the above screen where you may define the next child table for this parent table.

After you specify a child table name, press <RETURN>.

- 5. At the “Is this a new entry?” prompt, press <RETURN> to accept the “Yes” default and confirm that you are creating a new child table.

You see the Child Table Definition popup window, as shown below:

-----Base Table Definition-----	
Base Table Name	Description
Guides	
-----Characteristic Joins-----	
Parent Table	Child Tables (1/0) Editions
-----Child Table Definition-----	
References Table	Base Table Name: Editions
	Description:
	Use Default Structure? Yes_
	File name ^*parent
...Base Table Definition	
Press <PF1><PF3> For Help	

- At the Base Table Name field, press <RETURN> to accept the child table name you entered in the Child Tables field above.

You may change the name if you wish.
- At the Description field, you may enter an optional description of the child table.
- At the Use Default Structure? field, answer *Yes* or *No* to indicate whether or not you want the Data Dictionary to create a default physical structure for the child table.

Answer *Yes* to indicate that you want to use the default physical structure for your base table. Answer *No* to indicate that you want to custom-define the underlying physical structure for your base table. *Yes* is the default response.

- At the File Name field, press <RETURN> to accept the default automatically generated global file name, or specify another global file name. The default is either ^mdata (the default global), or ^*parent (which causes the data for the child table to be mapped as a subtree of the parent table's global structure). If you do not want the default, you can specify any global file name, new or existing. See "Child Table Default Master Map" on page 16-20 in Chapter 16, *Default Physical Structure*.

Note: To set the default global file name for the system, go to the Open M Developer and select the System Management menu option. From the System Management menu, select the System Configuration option. From the System Configuration menu, select SQL/Table Options. On this screen, set the "Use Hierarchical

Default Global Structure” option. Specify YES to set ^*parent as the default. Specify NO to set ^mdata as the default.

10. Press the <PROCEED> key to save your child table definition and exit the Child Table Definition popup window.

The Data Dictionary now creates the child table. If you answered *Yes* at the Use Default Structure? field, the Data Dictionary generates a skeletal default physical structure for the child table.

Later, you may complete the definition of your child table by selecting it from the list of existing base tables and then editing the definition to meet your requirements.

You return to the Characteristic/Designative Joins window.

To create another child table for this parent table, return to step 4 and repeat the procedure.

11. To exit the Joins window and return to the Base Table Definition main window, press the <PREVIOUS> key.

Defining a Designative Reference to Child Table

You may create designative reference fields that point from child-to-parent or from parent-to-child. However, you must create the child table before you can do this. To create a designative reference, both tables must already exist. A designative reference cannot hold a pointer to a table that may exist at some future time. When you create a child table, the Data Dictionary inserts an entry for the child table into the list of base tables defined in the current database. This allows you to use the child table in a designative reference. For further details, see “Defining a Designative Reference Field” on page 6-14 in Chapter 6, *Defining Base Table Fields*.

Generating Default Physical Structure

Before you can compile your base table or examine the default mapping structure, Open M with SQL must generate the default physical structure for your table. This structure defines the global, and the global subscripts to which each field in your table is mapped. As part of base table definition, Open M with SQL:

- n Creates the master map
 - Generates the Row ID field
 - Links each field to a global node
- n Creates index maps for unique fields and fields used as lookup fields.
- n Creates a skeleton default structure for child tables.
- n Stores the base table definition in the current M directory.

The operations are performed if you defined the base table with default physical structure. If you do not use default physical structure, you must explicitly define these items. See “If You Started Using Customized Physical Structure” on page 18-6 in Chapter 18, *Creating a Customized Map Definition*.

Edit the Default Global Name

Open M with SQL uses the default global ^mdata for all base tables that it maps. Each base table is given a separate node of that global. Child tables may take the default global ^*parent, which defines the child table’s global as a subtree within its parent table’s global.

For each base table, you can change the name of the global to which Open M with SQL maps the fields of that table. For further details, see “Changing the Global Name in a Default Physical Structure” on page 16-25 in Chapter 16, *Default Physical Structure*.

The global nodes are not created when the default physical structure is generated. They are created when you enter data into fields mapped to the node using Form Generator forms or SQL statements. Do not edit the Global Name once data has been entered, or existing data will be inaccessible.

Open M with SQL Generates Structure Automatically

If the default structure is not already defined for this table, Open M with SQL generates or recreates it automatically when:

- n You select the <Physical Structure> branching field at the Base Table Definition master window, followed by the <Examine Structure> branching field.

Open M with SQL will present the Create Default Structure window asking if you want Open M with SQL to create the default physical structure when:

- n You compile the base table
- n You exit the Base Table Definition window

Physical Structure Updates Automatically

Once the default physical structure is generated, it is automatically updated anytime the base table definition is edited and filed. You will see the following message on the message line:

```
Re-Calculating # Blocks in Each Map...
```

In addition, there may be messages about index maps and child tables. For further details about automatically generated physical structure, refer to “Examining a Default Master Map” on page 16-12 and “Examining a Default Index Map” on page 16-22 in Chapter 16, *Default Physical Structure*.

Row ID Field Created

When Open M with SQL generates the default physical structure, it creates a field with the data type Row ID. The name of the Row ID field is the initial name of the base table. Changing the base table name does not change the Row ID field name.

You cannot edit the definition of the Row ID field when you use default physical structure. However, you can examine its definition. See “Examining the Row ID Field Definition” on page 16-3 in Chapter 16, *Default Physical Structure*.

Procedure To generate the default physical structure:

1. Move your cursor to the last field on the Base Table Definition master window. Press <RETURN> to display the <Proceed> prompt.
2. Press <RETURN> at the <Proceed> prompt.

This displays a window containing the text: “Do you want to create the default structure for this table?”.

3. Press <RETURN> to accept the default response "Yes". Press <RETURN> again at the <Proceed> prompt.

Messages such as the following display sequentially on the message line:

```
Creating default structure for Docstaff...  
Re-Calculating # Blocks in Each Map...  
Creating map: Index Doc_Name 7  
Default structure created.
```

The Data Dictionary then displays the Base Table Definition row selection window, prompting you to define or select another base table.

Compiling a Base Table

Every base table must be compiled before a Form Generator form, or an SQL INSERT, UPDATE or DELETE statement that refers to it can be compiled successfully.

Compilation Produces Routines

Base table compilation produces executable routines that perform functions you defined along with the base table and its fields, including:

- n The Update Query: a compiled SQL query for retrieving the old values for a row, used during an UPDATE to tell which values have changed.
- n M code for validation checking and internal/external conversion.
- n Uniqueness Query(ies): one or more compiled SQL queries for each base table field specified as unique in the Data Dictionary, used to determine whether another row exists with the same value.
- n Compiled SQL queries for default Form Generator lookups, if defined, for rows of the base table.
- n A compiled SQL statement for each SQL-type base table trigger item.
- n Code for M triggers, if defined.
- n Filing code, if defined, for INSERT, UPDATE, and DELETE operations.

Base Table Routine Names

When you compile a base table, Open M with SQL creates one or more routines to contain the information defined in the table. For each table, a routine name prefix is used to produce the names of all of its associated compiled routines. The Data Dictionary provides a default prefix, which you can change.

To form a routine name, a single character is appended to the routine name prefix. The compiler uses the digits 1-9, followed by upper and then lower case alphabetic characters.

If you wish routine names to start with a prefix other than the default, *mt#*, then you must change the routine name prefix. You can change the routine name prefix by editing the base table's Advanced Options (see "Horizontal Options Menu" on page 5-8).

Don't Duplicate Existing Routine Names

Be sure no conflicting routine names exist, taking into account that each routine will have a digit or letter appended to it.

For example, if you define a prefix "rd", make sure there are no routines named "rd1", "rd2",... "rdZ",... "rdz".

Old Routines Automatically Deleted

When you recompile a base table, Open M with SQL deletes the old routines before beginning to compile. If the compilation is unsuccessful, there will be no base table routines for any forms or SQL INSERT, UPDATE or DELETE statements to use, so running them produces an error.

Edit the Routine Name before Recompiling

If you change the routine name prefix of a base table after it has already been compiled, and later recompile it, old routines whose names begin with the old prefix are not deleted automatically. You can delete them using the utility %urdel, as shown in the procedure below. Alternatively, you can go to the Macro Routine Utilities menu and select the Routine Delete option.

Example Given a base table with the default routine name prefix "mt5", the base table compiler appends consecutive single characters to form routine names for the base table:

```
mt51
.
mt59
mt5A
.
mt5Z
mt5a
.
mt5z
```

Routine Size

Machine requirements limit the maximum size of a single routine. If a base table requires more code than fits in one routine, Open M with SQL creates multiple routines for the base table. For compatibility across implementations, the maximum size of a single routine is 8KB.

Number of Routines

The number of routines generated during compilation depends upon the complexity of your base table. The more fields, lookups, indexes and triggers, the more routines. The number can vary from one to dozens.

Procedure To compile a base table right after defining it:

1. Enter the Base Table Definition master window.
2. Press <Go to Bottom Menu>.
3. Select the Compile Table option.

If old routines for this base table exist, they will be automatically deleted, unless you have just changed the prefix.

4. If you specified default physical structure when defining the base table, and Open M with SQL has not yet generated the default physical structure, the Data Dictionary displays a popup window that asks “Do you want to create the default structure for this table?”. Answer “Yes”. If you answer “No” at this prompt, or specified non-default physical structure, the Data Dictionary returns the message “Table *name* has no master map, cannot compile.”
5. Accept the default answer “No” at the prompt “Compile table in background”.

InterSystems recommends that you answer “No” so you can watch the compilation. This way, you know the compilation completed successfully or you see messages if errors occur. In addition, this method of compilation works on all versions of M. If you answer “No”, Open M with SQL displays the routines as it creates them during compilation.

If you answer “Yes”, the base table will compile transparently, so you can continue working on another Open M with SQL object. However, you will not know when compilation is complete, and you may not discover errors that occurred until you leave M.

Note: You cannot compile in the background when using Open M with SQL for DTM or DSM.

6. If you changed the routine prefix, you may want to delete old routines that have the old prefix by using the %urdel utility:

```
>DO ^%urdel
Routine(s): oldpfx*.*. *
Routine(s): <RETURN>
```

Compiling Related Objects

The Base Table Definition window horizontal options menu has an option for Compile Related Objects. Executing this utility creates a compilation configuration (using the Object Compilation Driver utility) for all objects related to the current base table. These related objects include the following:

- n The current base table.
- n The parent of the current base table
- n Queries that are defined via the Query Definition template and which name the current base table in the FROM clause
- n Queries that are defined via the Query Definition template and which name a view in the FROM clause that includes the current base table
- n Forms that use the current base table as their data source
- n Reports that use the current base table as their data source
- n Reports based on views that include the current base table in their join specifications

When you select the Compile Related Objects option, the following series of events occurs:

1. The utility finds all objects related to the current base table and inserts them into a temporary Object Compile Driver compilation configuration.
2. You can view and edit the compilation configuration just as you would if it were entered directly through the Object Compilation Driver utility. Editing the configuration lets you add objects to and delete objects from the list, suppress the compilation of objects, and view and/or edit the definition of any related object included in the list.
3. After you edit the configuration, press the <Proceed> key once to see a compile confirmation prompt. Answer *Yes* at the confirmation prompt, and press the <Proceed> key a second time to start the compile.
4. The utility compiles all objects in the configuration and then displays the results on the screen.
5. Before exiting, the utility prompts you with the option to permanently save the compilation configuration. To save the configuration, you must give it a name. If you save the configuration, you can later reaccess it using the Object Compile Driver utility.

Editing a Base Table Definition

Users with %ALTER privileges for a specific base table can edit its definition at any time. The owner of the base table automatically has %ALTER privileges. Along with the Open M with SQL System Manager, the owner can give other users %ALTER privileges to any owned base tables, using the Open M Developer main menu Privileges option.

You can modify the definition of a base table even if the globals to which it is mapped already contain data. This includes changing the name of the base table and adding and deleting fields.

Do Not Change Global Name Once Data Entered

Do not edit the value of your global name once data has been entered into base table fields.

You Cannot Change Status of a Child Table

Once you have created a child table from a parent table, you cannot change its status as a child table of a particular parent table.

If you decide the child table should not be a child table, but rather should be an independent table with a designative relationship to the parent table, then you:

- n Delete the child table and rebuild it
- n Remove the child table row from the Implicit Joins screen at the parent table

Pre-Delete Trigger Updated if Parent Reference Edited

If you edit the name of the parent reference field in a child table, which Open M with SQL creates automatically when you define a child table, Open M with SQL updates the default pre-filing delete trigger in the parent table, if it exists, to match the edited field name.

Procedure To edit a base table definition:

1. Enter the Data Dictionary.
2. Select the Base Table Definition option from the Data Dictionary menu.

This displays the Base Table row selection window.

3. Enter part of the name of the base table you wish to edit and press <Search Current Table>, or simply press <Search Current Table> at the blank field, to see a lookup box containing the names of all or matching currently defined base tables.

If you enter the complete name, go to Step 4.

You will see a reminder such as the following on the message line:

You must recompile the Guides table.

7. Recompile the base table.

See the section “Compiling a Base Table” on page 5-17.

8. Edit forms, reports, queries and views affected by the changes in this table.

For instance, if you delete a field, remove it from any report, query or view definitions that refer to it.

9. Recompile forms and reports based on this base table.

See the appropriate guide to learn how to edit and compile these objects:

- For forms, see the *Open M/SQL Form Generator Guide*.
- For reports, see the *Open M/SQL M/PACT Guide*.

Deleting a Base Table Definition

Procedure To delete a base table definition:

1. Enter the Data Dictionary.
2. Select the Base Table Definition option from the Data Dictionary Menu.
3. Select the base table you wish to delete at the Base Table Definition row selection window. (See Step 3 in “Editing a Base Table Definition” on page 5-21). Press <RETURN> to display the Base Table Definition window.
4. Press the <DELETE ROW> key.
5. At the "Do you want to delete this row?" prompt on the message line, type “Yes”, then press <RETURN>.

You see the message "<DELETED>" on the message line.

Restrictions on Deleting a Base Table Definition

Access Privileges Required

Users with the %ALTER privilege for a specific base table can delete its definition at any time. The owner of the base table automatically has %ALTER privileges. Along with the Open M with SQL System Manager, the owner can give other users %ALTER privileges to any owned base tables, using the Open M Developer main menu Privileges option.

Referenced Tables Cannot Be Deleted

You cannot delete a table that has a designative reference pointing to it.

- n A parent table cannot be deleted unless you first delete all of its associated child tables.
- n A designated table cannot be deleted unless you first delete (or change the data type of) the associated designative reference field in the designating table.

Attempting to delete a parent table with existing child tables displays the message: “Must delete dependents first:” followed by a list of the dependent child tables.

Associated Global Data is Not Deleted

If you delete a base table and the M global(s) represented by this base table contain data, you can no longer access this data via SQL, the Form Generator or M/PACT.

Caution: Deleting a base table definition does not delete the M global node(s) to which its fields are mapped or the data in those nodes.

To delete the M global data the base table represents, you must write an application to accomplish that task. Or, you may wish to write an SQL statement to delete the actual rows from the table. This can only be done before you delete the definition of the table.

Do Not Delete a Table Used in Views, Forms or Reports

Before deleting a base table, be certain it is not needed in an existing view, form or report. There will be unpredictable results if you do so.

Defining Base Table Fields

This chapter describes how to define the fields within a base table. Before defining base table fields, you must have created the base table and defined its physical structure, as described in Chapter 5, *Defining a Base Table*.

The topics covered in this chapter include:

- n Field Definition Overview page 6-2
- n Field Definition Checklist page 6-3
- n Steps 1-13 for Creating a New Field Definition
 - Steps 1 & 2: Specify a Field Name and Description page 6-4
 - Step 3: Specify the Data Type page 6-9
 - Step 4: Specify Maximum Length of Data page 6-33
 - Step 5: Specify If Values Must be Unique page 6-34
 - Step 6: Specify the Number of Distinct Values page 6-35
 - Step 7: Specify If the Field is Multi-Line page 6-37
 - Step 8: Define Computed Field Calculations page 6-42
 - Step 9: Define User Update Features page 6-46
 - Step 10: Define Data Conversion and Validation Code page 6-52
 - Step 11: Define Error and Help Messages page 6-52
 - Step 12: Define Additional Options page 6-52
 - Step 13: Save the Field Definition page 6-56
- n Copying a Field Definition page 6-57
- n Editing and Deleting a Field Definition page 6-59

Field Definition Overview

Once you have created a new base table and chosen whether to use default or customized physical structure as described in Chapter 5, *Defining a Base Table*, you are ready to define the database fields (also known as columns or attributes) in the table.

The maximum number of fields per base table depends on the internal representation of those fields. Numeric fields that use a conversion code (such as date fields) occupy more space internally than, for example, text fields. A maximum of 150 fields per base table is usually prudent. Increasing the size of the routine buffers on the system used for compiling and running the base table may permit a larger number of fields.

Caution: If you create a large number of fields in a base table, compilation may be slow and you may get <STORE> errors due to your partition size being too small.

You can define many kinds of attributes for each field. Most take defaults. Only a few of these attributes must be defined to create a working field definition.

Most parts of field definition affect data entry via SQL INSERT, UPDATE and DELETE commands, Form Generator forms, and M/PACT reports. However, some parts of the definition affect only forms (for example, Word Processing Capabilities for multi-line fields), or reports (for example, a Report Column Title).

This chapter contains a checklist and detailed steps that make it as easy as possible to define the fields in your base tables. To make most efficient use of this chapter:

1. Make copies of the Field Definition Checklist. Use this list to check off each step as you define each field in your base table.
2. For each step in the checklist, read the conceptual information and then follow the procedure to complete the step.

Every step in the checklist appears later in this chapter. For some steps, this chapter gives full details: conceptual information, examples, and a step-by-step procedure. Other steps refer you to other chapters in this Guide.

Field Definition Checklist

The Field Definition Checklist specifies all the required (Yes) and optional (Opt) steps in the field definition procedure. Optional steps that are highly recommended are designated (Rec).

Field Definition Checklist

	Step	Required?	Action
p	1.	Yes	Create a new field, assigning it a unique field name.
p	2.	Rec	Specify a text description of the field.
p	3.	Yes	Define the data type.
p	4.	Opt	Define maximum length. (All data types provide a default maximum length; changing this default is recommended for some data types, discouraged for other data types.)
p	5.	Rec	Define whether field is unique, meaning that no duplicate values may be input for this field.
p	6.	Rec	Specify number of distinct values.
p	7.	Opt	Define whether field is multi-line.
p	8.	Opt	Define whether field is computed.
p	9.	Opt	Select the Update Features option to define field protection level, whether a field value is required before the data for a row in this base table can be filed, and the default field value Open M with SQL uses when inserting a new row.
p	10.	Opt	Select the Conversion/Validation Code option to specify your own customized conversion and field validation algorithms, or edit those Open M with SQL automatically creates.
p	11.	Opt	Select the Error/Help Messages option to define your own help and error messages, including translations of these messages into other languages.
p	12.	Opt	Select the Additional Options option to specify the default column title to use in M/PACT reports.
p	13.	Yes	Save the field definition.

Steps 1 & 2: Specify a Field Name and Description

To create a new field, you need to give it a name and describe it.

Procedure To create a new field in a base table

- 1. Display the Base Table Definition main window for the base table.
- 2. Move the cursor to <Fields> and press <RETURN>.

Note: If this is a new table and the PROCEED SAVE Menu toggle is on, you will see the PROCEED SAVE Menu, with the highlight bar on the first option, SAVE. Press <RETURN> to save the current definition of this table. For information on the PROCEED SAVE Menu and toggle, see Chapter 3, *Using the Data Dictionary Interface* .

You will see the Field Definition row selection window. This is the row selection window for the Field Definition main window:

-----Base Table Definition-----+

Base Table Name

Guides_____

Description

List of all revisions to guides_____

-----Field Definition-----+

Field Name

Description

...Field Definition

Selecting

Press <PF1><PF3> For Help

The table below describes the fields on the Field Definition row selection window.

Table 6-1: Field Definition Row Selection Window Fields

Field Name	Description	Possible Values
Field Name	Required. Name must be unique within a specific base table, although a name can be reused in another base table. It is recommended that you make the field name unique across base tables by indicating in the field name the base table to which the field belongs. For example, "acct_date" for a date field in the Accounts table. If you later create a view from base tables which contain duplicate field names, you would be forced to rename all but one of the identically-named fields.	Same naming conventions as base tables: Can contain up to 40 alphanumeric and/or underscore characters. Cannot contain blank spaces or other punctuation characters. A name must begin with a letter. A name cannot be an SQL reserved word (listed in Appendix C, <i>SQL Reserved Words</i>). A name must be unique in its internal representation. Names are case-insensitive, and underscore characters are ignored when testing for uniqueness. Therefore, the Data Dictionary considers "ACCT_DATE", "acct_date", and "Acctdate" to be the same field name.
Description	Optional. If left blank, defaults to field name. This value is the default value used as a field caption or column header when you place this field on a form in the Form Generator or on a report in M/PACT. This value is overridden by any value you enter in the Report Column Title field when you choose the horizontal menu option Additional Options (see "Define Column Titles for Reports" on page 6-54). You can override this value in turn for a specific form in the Form Generator or a specific report in M/PACT.	Can contain up to 45 characters of any type, including blank spaces and punctuation characters.

3. Enter the name of the field and press <RETURN>.

Your cursor moves to the Description field.

4. Optionally, enter a description of the field and press <RETURN>.
5. You see the prompt "Is this a new entry?" on the status line. Press <RETURN> to accept the default value "Yes".

You see the Field Definition main window, with the cursor at Field Name. If you did not enter a description, the Data Dictionary copies the field name as a default description.

The Field Definition main window for the Title field of the Guides base table is shown below.

-----Base Table Definition-----

Base Table Name

Guides

Description

List of all revisions to guides

-----Field Definition-----

Field Name

Title

Description

Title of Guide

Data Type

Maximum Length

Unique Field?

Number of Distinct Values

Multi-Line Field? No

Computed Field? No

< Multi-Line Options >

...Field Definition

Unsaved Data

Press <PF1><PF3> For Help

Update Features

Conversion/Validation Code

Error/Help Messages

Indexing Options

Copy Field

Additional Options

Note the six horizontal menu options, and the message "Unsaved Data" on the status line at the bottom of this window.

Table 6-2 describes the fields and horizontal menu options on the Field Definition master window.

Table 6-2: Field Definition Master Window Fields and Horizontal Menu Options

Field Name	Type	Description	Default
Field Name	Text	You can edit the values you entered at the Field Definition row selection window.	None
Description			Field Name

Table 6-2: Field Definition Master Window Fields and Horizontal Menu Options

Data Type	Choose one: Date Designative Ref Multiple Choice Name Number Row ID Text Time Yes/No	Required. See “Step 3: Specify the Data Type” on page 6-9. Open M with SQL automatically defines a Row ID field under both default and customized physical structure. However, you can only modify this Row ID field definition when you are using customized physical structure. “Examining the Row ID Field Definition” on page 16-3 in Chapter 16, <i>Default Physical Structure</i> .	None
Maximum Length	Number, ranging from 1 to M string maximum length on your system.	Required. Serves as default for column widths in reports and queries and the data entry and display area in forms. You can override this value in the Form Generator and M/PACT. See “Step 4: Specify Maximum Length of Data” on page 6-33.	Based on data type.
Unique Field?	Yes/No	Yes means no two rows in the table may have the same value in this field. Provides automatic check for fields which should be unique, such as social security number. See “Step 5: Specify If Values Must be Unique” on page 6-34.	No
Number of Distinct Values	Text: number or NUMROWS	Approximate number of different values the field is expected to contain. Setting a realistic value here will help Open M with SQL applications using this field to run more efficiently. See “Step 6: Specify the Number of Distinct Values” on page 6-35.	None
Multi-Line Field?	Yes/No	Yes means the field can support unlimited length text fields. See “Step 7: Specify If the Field is Multi-Line” on page 6-37.	No
<Multi-Line Options>	Action Field	If you specified “Yes” as the Multi-Line Field option, this action field is activated. Moving the cursor to this field and pressing <RETURN> displays the Multi-Line Field Options popup window.	inactive
Computed Field?	Yes/No	See “Step 8: Define Computed Field Calculations” on page 6-42.	No
Update Features	Select this option to define: <ul style="list-style-type: none"> ▫ Field protection ▫ Whether a field is required ▫ Default value of a field See “Step 9: Define User Update Features” on page 6-46.		
Conversion/ Validation Code	Select this option to define field conversion and validation. See Chapter 10, <i>Data Conversion and Validation</i> .		

Table 6-2: Field Definition Master Window Fields and Horizontal Menu Options

Error/Help Messages	Select this option to define help and error messages, and their translations in other languages. See Chapter 13, <i>Field Help and Error Messages</i> .	
Indexing Options	Select this option to define an M expression that will be used by the Query Optimizer to select the optimal traversal path when the field is specified in an SQL query.	
Copy Field	Select this option to copy the definition of this field to create a new field with the same definition, either in this base table or in another existing base table. See “Copying a Field Definition” on page 6-57.	
Additional Options	See “Step 12: Define Additional Options” on page 6-52. One of these options permits you to define data column headers for M/PACT reports. See “Define Column Titles for Reports” on page 6-54.	

If you are importing a field, rather than defining it, refer to the table corresponding to the above in “Base Table Field Definitions” on page 20-7 in Chapter 20, *Importing Data Definitions* .

Step 3: Specify the Data Type

You can define the type of data that will be stored in a field in a base table. The data might be a date, text, name, number, multiple choice value, etc.

Open M with SQL data types save you a great deal of work. Open M with SQL only allows data of the type you select to be entered into the field. Open M with SQL automatically generates the following aspects of field definition, based on data type:

- n Conversion code, which you can modify.
- n Maximum length, which you can modify.
- n Internal validation code, which you can augment by entering additional validation code.

In addition, all data types except Yes/No have a pop-up auxiliary window where you can define additional attributes of that field type, such as a range of legal values or a format for a Date data type.

The table "Data Type Defaults" lists the available data types and their attributes.

Table 6-3: Data Type Defaults

Data Type	Valid Internal Values	Default Max Length	Number of Values	Conversion Code
Date	A 5 digit \$HOROLOG value	8-12, based on format	Variable	Yes
Designative Reference	A Row ID for the designated table	25	NUMROWS in designated table	External-to-Internal
Multiple Choice	User-defined	Longest choice	Num of choices, up to 1023	Yes, if internal values defined
Name	Lastname, Firstname [Middle] or Lastname Jr., Firstname Middle*	25	Variable	No
Number	A canonic number (+%val=%val)	16	Variable	Yes
Row ID	N/A	16	NUMROWS	No
Text	User-defined	30	Variable	No
Time	5 digit \$HOROLOG value	7	Variable	Yes

Table 6-3: Data Type Defaults

Yes/No	Y or N*	4	2	Yes
* See Chapter 10, <i>Data Conversion and Validation</i> , for details.				

Exercise Care When Mapping to an Existing M Database

If you are mapping a base table to an existing global structure, you must take great care when you assign a data type to a field. As the preceding table indicates, most of the Open M with SQL data types require specific internal storage formats. For example, a Date data type is always stored in a global as a 5 digit \$HOROLOG value. If an existing global value does not meet the format requirements listed, you must define the field linked to that global value as a Text data type. You then provide any needed conversion and/or validation code needed to complete the definition of that field.

Common Data Type Errors

Some common errors in data type assignment include the following:

- n Assigning a Date data type to a field which is stored in an alphanumeric format, such as 02JAN1999. Date fields imported from other database systems should not be assigned the Date data type.
- n Assigning a Time data type to a field which is stored in a clock-time format, such as 0830 or 8:30A. Time fields imported from other database systems should not be assigned the Time data type.
- n Assigning a Designative Reference data type to a field which is not stored as the Row ID value of the designated table, but as some other field or combination of fields in the designated table. For example, if you wish to designate a Doctors table, and the Doctors table Row ID is a numeric value, a field which is the doctor's name or license number would not be a valid Designative Reference. See “Examining the Row ID Field Definition” on page 16-3 in Chapter 16, *Default Physical Structure* .

Procedure To define a field's data type:

1. Move the cursor to the Data Type field.
2. Enter part or all of the data type name and press <List Choices>, or simply press <List Choices> at the blank field to see the data types available.

Data Type is a multiple choice field, so you will see a popup window displaying the list of available data types. The first type, "Date", is highlighted.

3. Move the highlight bar to the data type you want and press <RETURN>.

For all data types except Yes/No, an auxiliary window appears at which you define attributes specific to the data type. See the appropriate subsequent section in this chapter for the data type you are defining to learn how to complete the definition in the auxiliary window.

After you complete the auxiliary window definition, if any, the cursor moves to the Maximum Length field. Go to Step 4.

Defining a Date Field

You can define fields for calendar date values (month, day, and year) using the date data type. There is a separate Time data type for time of day.

Procedure To define a Date field:

1. Select "Date" in the Data Type field in the Field Definition master window.

The Date Data Type window appears:

Base Table Definition	
Base Table Name Guides	Description List of all revisions to guides

Field Definition	
Field Name Date_Retired	Description Date Guide Retired

Date - Data Type	
Date Format: 1_	Display: 10/18/66
First Date: _____	Display Full Year: No_
Last Date: _____	Override Display Delimiter: _
Formats: 1 - 10/18/90 5 - Oct 18, 1990 2 - 18 Oct 90 6 - October 18, 1990 3 - 18.10.90 7 - 90.10.18 4 - 18/10/90	

Up

...Date Data Type Unsaved Data Press <PF1><PF3> For Help

The table below describes the fields on the Date auxiliary window.

Table 6-4: Date Data Type Fields

Field Name	Description	Possible Values
Date Format	Determines how date will display. Default maximum length is set based on format.	1-7. Default is set as a System Configuration value. See bottom of Date Data Type window for examples of the available formats.

Table 6-4: Date Data Type Fields

Display	Shows an example of the currently selected date format.	The Display is modified by the Date Format value, the Display Full Year value, and the Override Display Delimiter value.
First Date	Earliest date which can be entered in this field.	Valid date. Optional, default is none (there are limits on the date imposed by the M storage format).
Last Date	Latest date which can be entered in the field.	
Display Full Year?	Should the year be shown as two digits or as a full four digits? All formats can display the year as a four-digit value; formats 5 and 6 always display the year as a four-digit value.	Yes/No Optional. Default is set as a System Configuration value.
Override Display Delimiter	Specify a character to be used as the delimiter when displaying the date. All date formats have a default delimiter, usually the backslash character or the blank space character.	Any single character. Optional, default is none

- Specify in the Date Format field the number of the date format desired. The default date format is set as a System Configuration value. To select a different format, press <Erase Field> to clear the Date Format field, then enter the number of the display format desired. An example of each date format is shown in the Formats portion of the Date Data Type popup window. The date format currently specified is shown in the Display portion of this window. This format governs how dates are displayed on forms and M/PACT reports, *not* how date data is stored within the database.
- Optionally, enter dates in the First Date field and the Last Date field which specify the earliest and latest date that can be entered in this field. You can enter these date values using any date format; Open M with SQL automatically converts them into the correct date format. The defaults are the \$HOROLOG limits: the years 1840 and 2114. You may also specify values for these fields by using the variable “T” (or “t”). T specifies today’s date; T-1 specifies yesterday’s date, T+1 specifies tomorrow’s date, and so forth. You

can specify any date for the First Date and Last Date, with the restriction that the First Date value must be earlier in time than the Last Date value. Changing the date format, full year, or display delimiter fields automatically changes the formatting of these fields.

4. Optionally, specify with the Display Full Year field whether the year should be displayed as two digits (for example, 97) or four digits (for example, 1997). The default is set as a System Configuration value. If Full Year=No, years are displayed as two digits for all formats except formats 5 and 6, which ignore the value of this field and always display the year as four digits. Changing this field automatically changes the values of the Display field, the First Date and Last Date fields, and the Maximum Length field. Changing this field does not effect the internal representation of date fields in the database.
5. Optionally, specify with the Override Display Delimiter field whether the delimiters separating day, month, and year should be displayed as shown in the Formats, or should be another character. You can specify any single character as a delimiter character. The default is to use the standard delimiter character for that date format. Changing this field automatically changes the values of the Display field, and the First Date and Last Date fields. It does not effect the Maximum Length field value.

The maximum length of a date field is automatically set by Open M with SQL, depending on the date format you selected. Do not make the length of a date field smaller than this default value.

Note: Date data imported from other database systems (for example, FileMan) cannot be stored in a Date data type field. See “Step 12: Define Additional Options” on page 6-52 for further details.

Internal Storage Format and Validation

Date values are stored in M internal date format, using a variable named \$HOROLOGY. The \$HOROLOGY variable consists of two 5-digit pieces, date and time, separated by a comma. The date portion is a count of days from December 31, 1840. Because of this, you may need to use a bit of sleight-of-hand to represent old historical dates, or dates in the remote future. Open M with SQL automatically generates validation code to verify this internal format. You cannot edit this validation code. However, you can provide additional validation code to augment this code.

User Entry Formats

Users can enter a date in any of the supported formats into a field of this type, unless the maximum length you specify is less than that needed by a particular format.

In addition, users can enter the variable T (or t) to represent the current system date. You can use T alone, or in an expression such as t-#, where # is a count of days. For example, t-1 is yesterday's date, and t+1 is tomorrow's date.

Conversion Code

Open M with SQL automatically generates both external-to-internal and internal-to-external code for a Date field. You can edit this code.

External-to-internal conversion code converts the date you enter into a \$H number.

Internal-to-external conversion code converts the \$HOROLOG value into the external date format you selected for the field.

If you are linking an existing M application to M with SQL through the data dictionary, the date data type should only be used for fields stored internally as the first comma piece of \$HOROLOG. If your application includes dates stored in another format, define them as text fields and use the conversion code facility, where necessary, to convert to external display values.

Defining a Designative Reference Field

Designative references are an Open M with SQL extension. They provide an implicit outer join between two tables. You create a designative reference with a field of data type Designative Reference. The designative reference field contains the Row IDs of rows in the designated table, which you specify. See Chapter 7, *Implicit Joins*, to learn more about designative references.

Internal Storage Value Must be Row ID of Designated Table

As mentioned earlier, if you are mapping to an existing M database you need to be certain to map a field of type Designative Reference to a global node which contains the Row ID of the designated table.

Designative References into Child Table

If your base table is a parent table and you want to create a designative reference field into one of its child tables, you must first create the child table by defining a characteristic relationship to it before creating the designative reference field that refers to the child table. This is because the designated table you name when you define a designative reference field must exist. See “Defining a Child Table” on page 5-11 in Chapter 5, *Defining a Base Table*, to learn how to define a characteristic relationship.

Procedure To define a Designative Reference field:

1. Select "Designative Reference" in the Data Type field in the Field Definition master window.

The Designative Reference data type auxiliary window appears. The completed window for the Latest_Edition designative reference field in the Guides table is shown below:

-----Base Table Definition-----	
Base Table Name	Description
Guides_____	List of all revisions to guides_____
-----Field Definition-----	
Field Name	Description
Latest_Edition_____	Des Ref to Editions Table_____
Data Type Designative Reference_____	
Maximum	ue Field? _____
+-----Designative Reference Data Type-----+	
Number	Referenced Table Editions_____
Multi	
+-----+	
Computed	
+-----+	
Update	Conversion/ Error/Help Indexing Copy Additional
...Designative Reference Data TypeUnsaved Data Press <PF1><PF3> For Help	

The table below describes the field on the Designative Reference auxiliary window.

Table 6-5: Designative Reference Data Type Fields

Field Name	Description	Possible Values
Referenced Table	Name of the designated base table that contains the field you want to reference from this table.	Name of an existing base table.

2. At the Referenced Table prompt, enter the name of the Base Table which contains the field you are referencing.

Defining a Multiple Choice Field

The multiple choice data type lets you supply a list of values for a field that will be displayed when a user presses <LIST CHOICES> when the cursor is on that field in a Form Generator form. Only the values you specify will be accepted for field assignment.

External and Internal Values

You can optionally define an internal value that is different from the display, or external, value. The internal value is the value stored in the database. If your external values are long, in order to provide clarity for application users, it often makes sense to define internal values which are shorter. Also, if you are mapping to an existing M database, some fields contain coded information which would make no sense to the user. In this case, you would develop meaningful external values for these fields.

NULL Internal Values

Starting with Version E, you can specify a NULL internal value. A choice with internal value NULL should be represented on the choice list as follows:

`<external_value>||`

If you do not explicitly define a default value for a multiple choice field and the user does not make a selection, that field will contain a NULL value. If the field has a NULL internal value choice, the external representation of the NULL choice will be displayed on a form or report once you insert a new row into the base table (via a form or via SQL).

Note: If you have existing database or form-only Multiple Choice fields with a NULL internal value, you must run the “Convert Multiple Choice Fields” conversion task in the D-to-E conversion program of the Conversion Manager in order to update their definitions.

Example You may want to store a shorter version of choices internally to save disk space:

ONE			1
TWO			2
NONE			

- The Multiple Choice data type auxiliary window appears. The completed window for the Product field in the Guides table is shown below:

[illegible]

The table below describes the fields on the Multiple Choice auxiliary window.

Table 6-6: Multiple Choice Data Type Fields

Field Name	Description
Do you want Internal and External Values?	Answer Yes if you want a different value to be stored in the database than the user will see and select in the multiple choice popup window.
Multiple Choices	Multi-line field. List one choice per line. You can list up to 1023 possible choices. A choice can be any alphanumeric string, up to 75 characters in length. The default maximum length of a multiple choice field is the length of the longest choice you supply. Do not make the length of a multiple choice field smaller than this value. If you are using both external and internal values, use the format: <i>External Value Internal Value</i>

2. Enter Yes at the "Do You want Internal and External Values?" prompt if you will provide internal as well as external values for the choices. Otherwise, press <TAB>.

3. At the Multiple Choice multi-line field, enter one choice per line, pressing <RETURN> after each choice. If you are entering both an internal and an external value for each choice, specify the values in the format:
external||internal.

The length of the longest (external name) choice you specify resets the default Maximum Length option. The number of choices you specify resets the Number of Distinct Values option for the field.

4. Press <PROCEED> when you are done entering choices.

Defining a Name Field

The name data type can be used for names that are stored in a particular internal format.

Internal Storage Value

The internal storage format for values in fields defined as Name data type can be one of the following:

Last Jr., First Middle

Last, First

Last, First Middle

Use the Name data type only when name values in your existing M database include both the first and last name, separated by a comma. The middle name or initial and title "Jr." are optional. If your application includes names stored in another format, (including something like title or rank, for instance), then define those fields as text fields and use the conversion code facility, where necessary, to convert to external display values.

Procedure To define a Name field:

1. Select "Name" in the Data Type field in the Field Definition master window.

The Name Data Type auxiliary window appears, with the cursor at the only field, "What Function..."

-----Base Table Definition-----

Base Table Name

Description

Guides

List of all revisions to guides

-----Field Definition-----

Field Name

Description

Name

Name of Staff Person

Data Type Name

Maximum

ue Field?

What Function of the Field should be Used
for Collating (ORDER BY, >) and Lookups

Exact

s >

Update

Conversion/

Error/Help

Indexing

Copy

Additional

...Function on Field in QueryUnsaved Data

Press <PF1><PF3> For Help

The table below describes the field on the Name data type window.

Table 6-7: Name Data Type Window Fields

Field Name	Description	Default Value	Possible Values
What Function...	Affects the data lookups retrieve	EXACT	ALPHAUP EXACT Minus Plus Space UPPER See "Collation Sequence Affects Name and Text Data Types" on page 6-28.

2.
- Press <RETURN> to accept the default collating function, EXACT. Or press <List Choices> to select one of the six types of collating available. Move the highlight bar to the collating function you want to use with this field, and press <RETURN>.

Defining a Number Field

Number data type parameters control how number values are displayed on forms, reports, and queries, and how number values can be inserted using forms or SQL.

Internal Storage Value

The value in a field defined as the Number data type must be in canonic form. Canonic means a purely numeric value, as opposed to a string value. This means it must follow this requirement:

$+ \% \text{ val} = \% \text{ val}$

Canonic numbers do not contain leading or trailing zeros. The Plus operator removes leading zeros, and trailing zeros that follow the decimal point. If the decimal point is followed by only zeros, eliminate the decimal point as well to make the number canonic. For example, the numbers 0123 and 1.0 are not canonic, since $+0123 = 123$ and $+1.0 = 1$.

Canonic numbers cannot contain any non-numeric characters, except the decimal point (when followed by a non-zero decimal fraction), the plus sign, and the minus sign. Commas may not be included in a canonic number. For example, the value 012ABC is not canonic, because $+012ABC = 123$. See “Plus” on page 6-30.

Validation and Conversion Code

Internal-to-external and external-to-internal conversion code is generated automatically for number data types. The actual code is dependent on the external number format you select for the field, as described in Step 6 of “To define a Number field:” on page 6-20.

See Chapter 10, *Data Conversion and Validation*, for more information.

Procedure To define a Number field:

1. Enter *Nu* or select *Number* from the popup window in the Data Type field in the Field Definition master window.

The Number data type auxiliary window appears. The cursor is on the Number Minimum field. The completed window for the Zip field from the Docstaff table is shown below:

```

+-----Base Table Definition-----+
Base Table Name      Description
Docstaff            Documentation Staff Info
+-----Field Definition-----+
Field Name      Description
Zip            Postal Zip Code
      Data Type Number
+-----Number Data Type-----+
Number Format: 1: 1234567.89
      1: 1234567.89      Decimal Places:
      2: 1234567,89      Leading Punctuation:
      3: 1,234,567.89     Show Minus Format: -x
      4: 1.234.567,89     Used in Arithmetic Calc? Yes
      5: 1 234 567.89     Number Minimum:
      6: 1'234'567,89     Number Maximum:
Up+-----+al
...Number Data Type      Unsaved Data      Press <PF1><PF3> For Help

```

The table below describes the fields on the Number data type window.

Table 6-8: Number Data Type Window Fields

Field Name	Description
Number Format	<p>Formats 1-6 (See Number Data Type window for an example of each format).</p> <p>The default number format used at your site is set as a System Configuration value. You may select any of the 6 available number formats for a particular field. This feature allows you to customize number formats for specific applications.</p> <p>You can change the format of a number data type field in both the Form Generator and M/PACT.</p>
Decimal Places	<p>Display: Exact number of decimal digits that should be displayed to the right of the decimal point.</p> <p>0 no decimal places displayed.</p> <p>blank number of decimal places stored internally will be displayed, up to column width format.</p> <p>1-11 number of decimal places displayed, up to column width.</p> <p>M/PACT allows you to override this value.</p> <p>Data Entry: Number of decimals the user may enter.</p> <p>0 no decimals are allowed.</p> <p>blank no checking for decimals is done.</p> <p>1-11 number of decimal places.</p>

Table 6-8: Number Data Type Window Fields

Leading Punctuation	Character to appear to left of field's first digit, such as \$.
Show Minus Format	Defines display format of negative numbers. -x minus sign precedes value, as in -999 x- Minus sign follows value, as in 999- (x) Negative value in parenthesis, as in (999)
Used in Arithmetic Calculation?	Choose Yes (default) for fields which may be used for calculations, such as salary or cost. Choose No for fields like Social Security Number or Account Number.
Number Minimum	Lowest number that can be entered.
Number Maximum	Highest number that can be entered.

2. At the Decimal Places field, enter the number of decimal places the user is allowed to enter and press <RETURN>.
3. At the Leading Punctuation field, enter the character you want to precede numbers when they are displayed.
4. At the Show Minus field, enter -x, x- or (x) to indicate how to display negative numbers.
5. At the Used in Arithmetic Calculations? field, press <RETURN> to accept the default value Yes if the field is used in calculations or enter *N* if the number is not used in calculations.
6. At the Number Format field, press <RETURN> to accept the default format, or press <Erase Field Line> and enter the number of the format you want.
7. At the Number Minimum field, enter the smallest number the user is allowed to enter on a form and press <RETURN>.
8. At the Number Maximum field, enter the largest number the user is allowed to enter on a form and press <RETURN>.
9. Press <RETURN> at the <proceed> prompt.

Defining a Row ID Field

When you create a base table, the Data Dictionary automatically creates a Row ID field for the table, sets its field name as the base table name and its field definition options as shown on the following screen. Row ID field definition is completely automatic.

-----Base Table Definition-----					
Base Table Name		Description			
Accounts_____		Client accounts master table_____			
-----Field Definition-----					
Field Name		Description			
Accounts_____		Accounts Row ID_____			
Data Type		Row ID_____			
Maximum Length		16_		Unique Field? Yes_	
Number of Distinct Values		NUMROWS_____			
Multi-Line Field? ____		< Multi-Line Options >			
Computed Field? No__					
...Field Definition					
Inquiry Mode		Press <PF1><PF3> For Help			
Update Features	Conversion/ Validation Code	Error/Help Messages	Indexing Options	Copy Field	Additional Options

You cannot modify any of these field definition values; however, you can define additional Row ID data type options by moving the cursor to the Data Type field and pressing <RETURN>. These options are further described in “Examining the Row ID Field Definition” on page 16-3 in Chapter 16, *Default Physical Structure*, and in “Step 3: Edit Row ID Field(s)” on page 17-10 in Chapter 17, *Relational Definition of an M Database*.

Defining a Text Field

The text data type can be used for any ASCII text string.

Procedure To define a Text field:

- 1. Select "Text" in the Data Type field in the Field Definition master window.

The Text data type auxiliary window appears. The user has pressed <List Choices> so you see the function lookup box:

-----Base Table Definition-----

Base Table Name

Description

Guides_____List of all revisions to guides_____

-----Field Definition-----

Field Name

Description

Title_____Title of Guide_____

Data Type Text_____

Maximum_____ue Field? _____

What Function of the Field should be Used
for Collating (ORDER BY, >) and Lookups

Exact_____s >

Update

Conversion/

Error/Help

Indexing

Copy

Additional

...Function on Field in QueryUnsaved DataPress <PF1><PF3> For Help

The table below describes the fields on the Text data type window.

Table 6-9: Text Data Type Window Fields

Field Name	Description	Default Value	Possible Values
What Function...	Affects the data lookups retrieve	EXACT	ALPHAUP EXACT Minus Plus Space UPPER See "Collation Sequence Affects Name and Text Data Types" on page 6-28.

- 2. Press <RETURN> to accept the default collating function, EXACT. Or press <List Choices> to see the six types of collating available. Move the highlight

6-24 Open M with SQL Data Dictionary Guide

bar to the collating function you want to use with this field, then press <RETURN>. See “Collation Sequence Affects Name and Text Data Types” on page 6-28.

Defining a Time Field

The time data type is for time of day values. Internal-to-external conversion code is generated automatically.

Internal Storage Value

The value enter in Time data type fields are store as the second piece of the M \$HOROLOG function, which is the number of seconds since midnight.

User Entry Formats

Users can enter a time in any of the supported formats into a field of this type, unless the maximum length you specify is less than that needed by a particular format.

In addition, users can enter the variable N or n to represent the current system time. You can use N alone or in an expression, such as N-#, where # stands for the number of seconds. For example: N-1 means the current time minus 1 second.

Note: Time data imported from other database systems (for example, FileMan) cannot be stored in a Time data type field. See “Step 12: Define Additional Options” on page 6-52 for further details.

Validation and Conversion Code

Validation code is created from the values you enter in the First Time and Last Time fields. If you need to base permissible values on relative times (such as disallowing any times prior to the actual calendar time), leave these fields blank and add appropriate field validation code.

Internal-to-external and external-to-internal conversion code is generated automatically for Time data types. The actual code is dependent on the external time format you select for the field, as described in Step 2 of the “To define a Time field:” on page 6-25.

Procedure To define a Time field:

1. Select "Time" in the Data Type field in the Field Definition master window.

The Time data type auxiliary window appears:

```

+-----Time Data Type-----+
|                               |
|      Time Format 1. 04:59PM__|
|                               |
| First Time _____ Last Time _____|
|                               |
|      Formats:                |
|      1 - 04:59PM      3 - 04:59:59PM|
|      2 - 16:59       4 - 16:59:59|
|                               |
+-----+

```

The table below describes the fields on the Time data type window.

Table 6-10: Time Data Type Window Fields

Field Name	Description	Default Value	Possible Values
Time Format	Display format of data in field. Determines maximum length.	1	1-4 See window.
First Time	Earliest time that can be entered.	None	Any valid time format, or n to represent current time at data entry.
Last Time	Latest time that can be entered.	None	

- Specify in the Time Format field the number of the time format desired. The default is time format 1. To select a different format, press <Erase Field> to clear the Time Format field, then enter the number of the display format desired. An example of each time format is shown in the Formats portion of the Time Data Type popup window. The time format currently specified is shown at the top of this window. This format governs how times are displayed on forms and M/PACT reports, *not* how time data is stored within the database.
- Optionally, enter times in the First Time field and the Last Time field which specify the earliest and latest time that can be entered in this field. You can enter these time values using any time format; Open M with SQL automatically converts them into the time format you specified. Open M with SQL assumes that a time value is AM, unless you specify otherwise. Midnight is represented as 00:00:00 (not 24:00:00). You may also specify values for these fields by using the variable “N”. N specifies the current time (N=Now); N-1 specifies one second ago, N+1 specifies one second from now, and so forth. You can specify any time for the First Time and Last Time, with the restriction that the First Time value must be earlier than the Last Time value.

Changing the time format automatically changes the formatting of these fields.

4. Press <RETURN> at the <proceed> prompt.
5. The default maximum length value is set by Open M with SQL, based on the time format you selected. Do not decrease the maximum length below this value.

Defining a Yes/No Field

When you create a Yes/No field, no auxiliary window appears.

Internal Storage Value

Whether the user enters *Yes*, *No*, *Y*, or *N*, the internal value stored is Y for a Yes value and N for a No value. If no value is entered, there is a null value.

Null Internal Values

Sometimes you need to allow for null values, as when certain information may be unknown at the time of initial data entry. In these cases, your application must take the existence of null values in these fields into account, by considering three values in the field: Y, N and NULL.

No Null Internal Values

Some applications do not need to provide for null values. To make the existence of null values impossible in order to simplify application code, make the Yes/No field required and include a default value for ease of use.

Example In the Guides table, we always want to know whether a title is retired or active. Therefore, we do not allow NULL values. The Update Features window for this

field is shown below, where we define it as required and give it a default value of No. See “Step 9: Define User Update Features” on page 6-46:

-----Base Table Definition-----		
Base Table Name	Description	
Guides_____	List of all revisions to guides_____	
-----Field Definition-----		
Field Name	Description	
Retired_____	Is Title Retired?_____	
-----Update Features-----		
Field Protection Add/Edit_____	Column Number 3	Field ID # 45
Required Field? No_____ If it is 'Maybe' Required - the M Expression to Determine if it is Required is: IF _____		
M Expression for Default (External) Value _____		

...Update Features Unsaved Data Press <PF1><PF3> For Help		
Conversion/ Validation Code		Error/Help Messages

Collation Sequence Affects Name and Text Data Types

Collation sequence is an ordering syntax defined on data types which are character strings, such as Name and Text.

There are six available collating sequence functions in Open M with SQL: EXACT, ALPHAUP, Minus, Plus, Space, and UPPER.

Numbers Come First

In all six collation sequences, pure numbers come first in numeric order, followed by all other strings. A number that is not canonic (e.g., 02 or 1.30) collates as a string rather than a number, unless you are using the Plus collating sequence.

EXACT

When you define a field of type Name or Text in the Data Dictionary, EXACT is the default collation sequence.

EXACT is an M collation sequence that orders pure numeric values (values for which x=+x) in numeric order first, followed by all other characters in string order. The M collation sequence for strings is the same as the ANSI-Standard ASCII collation sequence--except on IBM mainframes, where it is the EBCDIC collation sequence. In ASCII collation sequence, digits are collated before uppercase alphabetic characters and uppercase alphabetic characters are collated

before lowercase alphabetic characters. Punctuation characters occur at several places in the sequence.

Example The following example shows several strings listed in default (EXACT) collation sequence. Note that the apostrophe character collates before 'A'.

String
A'Ha
ARNOLD
Adams
a'Choo
aaronson

ALPHAUP

If the ALPHAUP collation sequence is used, the strings to be compared are first converted to ALPHAUP format and then compared according to EXACT collation sequence criteria. ALPHAUP format removes all punctuation (non-alphanumeric) characters except commas and question marks and translates all alphabetic characters into uppercase.

Example This example shows the same strings listed in ALPHAUP collation sequence. They are converted as shown in the "Compared As" column before being used in an operation.

String	Compared As
aaronson	AARONSON
a'Choo	ACHOO
Adams	ADAMS
A'Ha	AHA
ARNOLD	ARNOLD

Minus

This collating sequence reverses the collating order for numeric values. The index map sequence is `-{field}`.

Plus

This collating sequence deletes non-canonic characters from numbers. These include:

- n Leading zeros (for example, 0777)
- n Trailing zeros (for example, 777.10)
- n Decimal points, when the decimal fraction=0 (for example, 777.00)
- n Commas (for example 7,777)
- n Other non-numeric characters (except +, -, and .) (for example, \$20.98)

This causes numeric values that were input with these characters to be collated as numbers, rather than as character strings. If the Plus collating sequence is not used, non-canonic numbers are collated as strings.

The index map sequence is +{field}.

Space

This collating sequence forces all values to collate as character strings. The index map sequence is “_{field}.

UPPER

If UPPER collation sequence is used, the strings to be compared are first converted to UPPER format and then compared according to EXACT collation sequence criteria. UPPER format translates alphabetic characters into uppercase but leaves punctuation characters intact.

Example Here are the same strings listed in UPPER collation sequence.

String	Compared As
a'Choo	A'CHOO
A'Ha	A'HA
aaronson	AARONSON
Adams	ADAMS
ARNOLD	ARNOLD

Effects of Collation Sequence

Collation sequence affects the following operations:

- n The order produced by an SQL ORDER BY clause
- n The behavior of the comparison operations
 - > greater than
 - < less than
 - = equal to
 - >= greater than or equal to
 - <= less than or equal to
 - not> not greater than
 - not< not less than
- n The behavior of form lookups

Collation Sequence and ORDER BY

ORDER BY applied to an expression that is a field with a data type of Text or Name uses the collation sequence defined for that field in the Data Dictionary.

ORDER BY applied to any other expression uses the EXACT collation sequence.

Example If the EmpName field has the collation sequence ALPHAUP, this example orders the returned rows first by "DateEnd-DateStart" in EXACT collation order, and within that by "EmpName" in ALPHAUP order:

```
SELECT DateEnd-DateStart, EmpName ...
ORDER BY 1, 2
```

Collation Sequence and Comparisons

In general, comparisons of character string values are done using EXACT collation sequence.

If either side of a collation operator is a field with a data type of Text or Name, the comparison uses the collation sequence defined for the field in the Data Dictionary. For example, if the EmpName field is defined as ALPHAUP, the comparison:

```
EmpName > "jo"
```

tests EmpName in ALPHAUP format to determine whether it is greater than (follows in collation sequence) the string "JO". In effect, an "alphaup function" is

implicitly applied to both sides to convert each to ALPHAUP format. Then, the (EXACT) comparison is performed.

If both sides of a comparison are fields of Text or Name data type, precedence order is used to choose the collation sequence of the comparison. The precedence order is:

1. ALPHAUP (highest)
2. UPPER
3. EXACT (lowest)

The collation sequence with the higher precedence is chosen.

Collation Sequence and Form Lookups

Lookups affect how users can retrieve rows from a base table from a Form Generator form. See “Case Transformation” on page 8-16 in Chapter 8, *Base Table Lookups*, for information on how collation sequence affects lookups.

Step 4: Specify Maximum Length of Data

Maximum length specifies the longest value (in number of characters) that can be stored in this field. Open M with SQL specifies a default value for the Maximum Length, based on the data type you selected. Do not edit this default value for data types Date, Time, Multiple Choice, or Yes/No.

Procedure To edit the default maximum length:

1. Move your cursor to the Maximum Length field on the Field Definition master window.
2. Press <Erase Field>, then enter the correct value for your application. The maximum value permitted is 999. Maximum length specifies the length of a single line of a multi-line field (see “Step 7: Specify If the Field is Multi-Line” on page 6-37 for further details).

Step 5: Specify If Values Must be Unique

If a field should have a different value for each row in the database, define that field as unique. For example, in a table of Employees an employee ID number field would be a unique value; an employee name field, though unlikely to have many duplicates, is not by definition unique, and therefore should not be defined as unique.

Open M with SQL ensures that a value defined as unique does not already exist in another row. If a user enters a value which already exists, they see an error message and must re-enter a unique value.

When Open M with SQL defines the Row ID field for a table, which it does whether you are using default or customized physical structure, it defines that field as unique.

Index Maps for Unique Fields

If you use default physical structure, Open M with SQL automatically creates an index map for unique fields. If you use customized physical structure, InterSystems urges you to define an index map for each unique field (other than the Row ID field), as described in “Unique Fields” on page 9-4 in Chapter 9, *Index Maps*.

Procedure To define a unique field:

1. Move your cursor to the Unique Field? prompt on the Field Definition master window.
2. Enter Yes if you know that each row should have a different value for this field.

If you are using customized physical structure and the data type of this field is Row ID, you must answer *Yes* unless the Row ID is a combination of field values.

Example In the Guides table, we defined the Title field as unique to ensure no duplicate entries.

Step 6: Specify the Number of Distinct Values

The Number of Distinct Values field asks for the approximate number of different values the field is expected to have in the database. For example, a Yes/No field has 2 distinct values if it is required; otherwise, it has 3 distinct values, since it can also be NULL.

The Open M with SQL Query Optimizer uses the number you enter in this field to select a retrieval strategy. The number need not be precise; "ballpark figures" (like 20, 2000, or 200000) are acceptable. Enter the number without commas or other punctuation.

The value in this field can be one of two possibilities:

- n A number
- n The reserved word NUMROWS

Enter a Number

For some fields, the Number of Distinct Values is a constant that is independent of the size of the database. For a multiple choice field, M with SQL adjusts this field to the number of choices you specified. For other fields, you should specify a value, if possible, to reduce processing overhead. For example, a whole percentage field would only have 100 possible values. A days remaining in year field would have 366 (including zero) possible values.

Enter NUMROWS

For some fields, the Number of Distinct Values is a function of the number of rows in the base table. Whenever you define a field as unique, the Number of Distinct Values is always equal to the number of rows in the base table. Non-unique fields, such as Phone in the Docstaff table, may not be unique; however, the number of distinct values is apt to be close to the number of rows in the table since duplications, while possible, are unlikely.

You use the notation NUMROWS to refer to the number of rows in the base table. The value of NUMROWS is equal to the number you supplied as the Approximate Number of Rows in the base table definition. (See Table 5-2: Fields on Base Table Definition window, on page 5-6 in Chapter 5, *Defining a Base Table* .)

You can also use the syntax:

```
NUMROWS/any number greater than 1
```

as in:

NUMROWS/2 or NUMROWS/50 or NUMROWS/1000

Use Rows in Designated Table for Designative Reference Fields

If you are specifying the Number of Distinct Values for a designative reference, enter the number of rows in the designated table.

Example The Guides table includes the Designative Reference field, Latest_Edition_Gui. For Number of Distinct Values, we use the number of rows we estimate will be in its designated table, Editions.

Procedure To specify the number of distinct values:

1. Move your cursor to the Number of Distinct Values field on the Field Definition master window.
2. Enter the number that estimates the number of different values that might appear in this field, or enter the word *NUMROWS* if the field is unique or there is apt to be a different value for each row.

If you did not define your field to be unique, your cursor will move to the "Multi-Line Field?" field; see "Step 7: Specify If the Field is Multi-Line" on page 6-37. If your field is unique, your cursor will move to the "Computed Field?" field, since multi-line fields cannot be defined as unique; see "Step 8: Define Computed Field Calculations" on page 6-42.

Step 7: Specify If the Field is Multi-Line

A multi-line field is a database field which spans more than one line. It provides maximum flexibility to the developer, allowing users to:

- n Enter text of unlimited length

For example, the Author_Role field in the Editions table allows paragraph style description of the tasks performed by the main author of an edition.

- n Enter a list of related items which are always managed as a unit, one item per line.

For example, the Summary field in the Edition table may contain many new features covered in the Edition.

The maximum length of a multi-line field refers to the maximum length of one line.

The maximum number of lines in a multi-line field is limited to the maximum number of array nodes allowed in your version of M.

Displaying Multi-Line Fields on Forms and Reports

The Form Generator and M/PACT allow you to choose how many lines of a multi-line field to display, with the range being from 2 to 22 lines. If there are more lines of data than lines displayed, users can scroll through the remaining lines. A display indicator tells you what line the cursor is on and the total number of lines in the field.

Multi-Line Fields May be Computed Fields

Open M with SQL supports multi-line computed fields. This is especially useful if you have an existing application which uses a form of multi-line storage different from the two types described in "Two Storage Modes" below. See the discussion in "Step 8: Define Computed Field Calculations" on page 6-42.

Two Storage Modes

Multi-line fields can be stored in one of two ways:

- n One Line Per Node
- n All in One Node

When you use default physical structure, Open M with SQL automatically uses One Per Node storage mode. You cannot alter this.

When you use customized physical structure, Open M with SQL displays an auxiliary window where you can select one of the two storage modes.

Note: If your existing M database does not use either storage type, you can define its storage structure at the Map Definition form, at the <M Retrieval Code> window reached from the Map Data Specifications window.

One Per Node Storage Mode

This is the mode used under default physical structure. In One Per Node storage mode, each line of a multi-line field is stored in a separate node. The first subscript level down is given an integer value, starting with one for the first multi-line field in a table. The value of this node is the number of lines in the field. Each line is stored one subscript level down; the value of the subscript is an integer, starting with 1 for each new multi-line field. The maximum length of a line is the maximum M string length allowed in your version of M.

Example The Author_Role and Summary multi-line fields in the Editions base table is stored using One Per Node storage mode, since the table is created with default physical structure. This is how the corresponding global looks:

```
^Editions(1,1,1,0)=2;Author_Role field storage
  ^Editions(1,1,1,1)=Editing
  ^Editions(1,1,1,2)=Chapters 1,2,3,6,7,8,11,12
  ^Editions(1,1,2,0)=4      ;Summary field storage
  ^Editions(1,1,2,1)=Customized Mapping
  ^Editions(1,1,2,2)=Hypertext Help
  ^Editions(1,1,2,3)=Import File Specifications
  ^Editions(1,1,2,4)=Data Dictionary Reports
```

See Chapter 16, *Default Physical Structure* , to see the corresponding definition of these fields at the Map Data Specifications screen.

All in One Node Storage Mode

If you are using customized physical structure, you can choose **All in One Node** storage mode. This mode stores a list of items as pieces of a node value, each item separated by a delimiter. A disadvantage of this mode is that the length of the entire field is limited by the maximum global reference length and the maximum M string length allowed in your version of M.

Example Here is how the Author_Role and Summary multi-line fields would be stored in a global if you were using customized mapping and chose All in One Node storage mode:

```
^Editions(1,1,1)=Editing^Chapters 1,2,3,6,7,8,11,12
^Editions(1,1,2)=Customized Mapping^Hypertext Help^Import File
                  Specifications^Data Dictionary Reports
```

See Chapter 18, *Creating a Customized Map Definition*, to see how to define Map Data Specifications for a multi-line field using One Per Node storage mode.

Referencing Multi-Line Fields

You can reference multi-line fields when you insert M code as part of your customizing of base tables and forms. You use the same syntax, whether you are using One Per Node or All In One Node storage mode.

The syntax to reference the counter of the number of lines is:

```
{fieldname()}
```

Open M with SQL manages this counter automatically. It stores the counter in the first multi-line global node for One Per Node storage; it manages it internally in memory for All in One Node storage.

The syntax to reference an instance of the multi-line field is:

```
{fieldname(expression)}
```

Example If you manage updating the multi-line field {Author_Role} in M code, you would write the following code to add a new line to the field:

```
SET n={Author_Role()+1}          Determine number of next line
SET {Author_Role(n)}="Chap 5"    Insert value in the node of that number
```

Automatic Repacking

Open M with SQL automatically repacks the multi-line field after computation, eliminating all null instances, and then resets the counter to the revised actual number of instances. Therefore, if you issue a command to set an instance, and the value it receives is null, that instance will be removed from the multi-line field.

Set Field to Null

You can set either all lines or one line of a multi-line field to be null. If you want to nullify the entire field, issue the command:

```
KILL {fieldname()}
```

You can also nullify the entire field by using the IF conditions in the Computed Field auxiliary window.

Procedure To define a multi-line field:

1. Enter Yes in the Multi-Line Field? field on the Field Definition master window. This displays the Multi-Line Field Options window:

-----Base Table Definition-----+

Base Table Name

Guides_____

Description

List of all revisions to guides_____

+-----Multi-Line Field Options-----+

Field Name

Street_____

Data Type Text_____

Maximum Length 30_

Number of Distinct Va

Multi-Line Field? Yes

Computed Field? No_

Do you want Word Processing Capabilities
When editing this Multi-Line Field?

No_

Hard Return Indicator: ~

Display (Current Line/Total Line) Indicator?

Yes_

Maximum number of lines to allow _____

Storage Type

One per Node_____

Storage Delimiter

Update

Conversion/

Error/Help

Indexing

Copy

Additional

...Multi-Line Field OptionsUnsaved Data

Press <PF1><PF3> For Help

The table below describes the fields in the Multi-Line Field Options window.

Table 6-11: Multi-Line Field Auxiliary Window Fields

Field Name	Description
Word Process- ing Capabilities	Yes or No. Default is No. “Yes” enables automatic line-wrapping, enhanced field navigation, and the ability to perform string searches. For further details on word processing capabilities, see the <i>Open M/SQL User Interface Programming Guide</i> . “No” halts typing at the end of each line; a <RETURN> is required to go to the next line.
Hard Return Indi- cator	Single character that indicates a forced line return/start of new line.
Display Indicator	Yes or No. Default is Yes. “Yes” displays a line counter that indicates the current line number/total number of lines. For example, 2/3 indicates the current line is the second of three lines. “No” suppresses the display of the line counter on forms at runtime.

Table 6-11: Multi-Line Field Auxiliary Window Fields

Maximum Number of Lines to Allow	<p>Specify the maximum number of lines to allow for storage in this multi-line field. This maximum number of lines applies even if this field is defined as a word-processing (line wrapping) multi-line field. At the form definition level you can override this value to decrease the maximum number of lines allowed; you cannot increase this number at the form definition level. If you specify no value, an unlimited number of lines are allowed.</p> <p>Open M with SQL rejects SQL INSERT and UPDATE statements if they contain a multi-line field with a line count that exceeds this value. This value only limits data input, it does not limit the maximum number of lines to display.</p>
Storage Type	<p>This option is only modifiable for customized physical structure. For default physical structure, One Per Node is always used. One Per Node is the default for customized physical structure.</p> <p>If you are using the field to store unlimited-length text, select All in One Node.</p> <p>However, if you are choosing to store a list of related items, such as an address, you may select either type.</p>
Storage Delimiter	<p>This field is required if you select the storage type, All in One Node. Enter:</p> <ul style="list-style-type: none"> ▫ Actual delimiter character in quotes, as in "/" or ▫ Ascii value, using the M \$CHAR function as in \$C(18). <p>If you enter "^", a list of colors would be stored in a node as:</p> <p style="text-align: center;">red^blue^green^purple</p>

2. At the Storage Type prompt, press <List Choices> to see the possible values or enter the first letter of the type you want and press <RETURN>.

If you select All in One Node, your cursor will go to the Storage Delimiter field; see Step 3. Otherwise, go to Step 4.

3. Enter the Storage Delimiter.
4. Press <PROCEED>.

The cursor now goes to the Computed Field? field.

Step 8: Define Computed Field Calculations

Computed field values are calculated by M expressions that can reference other fields in the same base table. The computation can reference a maximum of approximately 35 other fields.

Computed field calculations can also include M \$ variables such as \$H (date/time) and \$ZD. A computed field value can be simply a calculation that is performed when the value is needed (at M/PACT report, query, or form run time), or the values of a computed field can be stored in the database and used in the map definition. You can specify that computed field values in the current row should be recalculated every time a value changes for any field that is used in the computation.

M code for computing the field's value is specified within the Computed Field definition. The field data type determines which values are allowed for the field, with the same restrictions that apply for a field of that data type.

Benefits and Limitations of Computed Fields

Computed fields are useful if you wish to have a default value based on another field, or if you wish to assign a default value at times other than upon the insertion of a new row in the table.

A computed field may be multi-line. Use the multi-line syntax described in “Step 7: Specify If the Field is Multi-Line” on page 6-37.

A computed field definition may not be copied. See “Copying a Field Definition” on page 6-57.

Two Types of Computed Fields

There are two types of computed field, those which are:

- n Always equal to the result of the computation
- n Not always equal to the result of the computation

Always Equal to Computation

Values for a field of this type are not stored in the database.

If you select this type in the Computed Field auxiliary window, follow these guidelines:

- n Set field protection to Output Only.
- n If you are using customized mapping, do not map this field to a global.

If you select this field as a lookup display field, no value will appear during row selection.

Computed fields of this type are validated based on the final result of the computation

Example In the Docstaff table, the field Length_of_Service is computed based on the value of the Date_Of_Hire field. We designate it as Always Equal to the Computation as there is no need to store this value in the database.

Not Always Equal to Computation

Values for this type of computed field are stored in the database.

If you select this type, the following applies:

- n It does not have to be output only.
- n If you are using customized physical structure, you must include it on the Master Map so it can be stored. Open M with SQL will do this automatically if you are using default physical structure.
- n It will be computed as follows:
 - When inserting a row
 - If its old value is null
 - If certain fields you specify on the Computed Field auxiliary window are updated
- n It will not be computed under the following circumstances:
 - If it is to be overridden by an explicit value
 - When updating a row, if it has a previous value, and none of the fields used to trigger the computation is updated
 - During a lookup, if you selected it as a field to be displayed

Example In the following figure, the Total field of the Line Item table is computed, based on the Quantity and Unit Cost fields:

Item Name	Quantity	Unit Cost	Total
Chair	2	\$150.00	\$300.00
Bed	1	\$600.00	\$600.00
Table	2	\$100.00	\$200.00
Couch	1	\$1000.00	\$1000.00
Dining Table Chair	4	\$125.00	\$500.00
Lamp	6	\$50.00	\$300.00
Rug	4	\$200.00	\$800.00
Bookcase	4	\$80.00	\$320.00

Any change to the value of either the Unit Cost or Quantity field changes the Total. However, since this is financial data, you want to be sure there is a record of it on disk, so you define it as Not Always Equal to the Computation.

Procedure To define a computed field:

- 1. Enter “Y” at Computed Field? at the Field Definition master window.

You will see the Computed Field auxiliary window:

-----Base Table Definition-----+
Base Table Name Description
Guides List of all revisions to guides
-----+-----Computed Field-----+
Code to compute the result: (1/1)
 (Explicitly set {*} to the computed field value.)
SET {Employee_Date}=Date_of_Hire/365.25

Exception: The computation produces a NULL result
 if any of the following is true:
 (1/0)
 IF Date_of_Hire=" " _____
 IF _____
 IF _____

Is the value always equal to the computation? Yes_
-----+-----
...Computed Field Unsaved Data Press <PF1><PF3> For Help
Update Conversion/ Error/Help Indexing Copy Additional
Features Validation Code Messages Options Field Options

The table below describes the fields on the computed field window.

Table 6-12: Computed Field Auxiliary Window Fields

Field Name	Description
Code to compute the result:	Enter any valid M expression, including embedded SQL (##sql or &sql). Refer to the computed field itself with {*}. You can refer to other fields, in the same base table only, by using the field name surrounded by curly braces, as in {fieldname}. You can refer to a line in a multi-line field by using the form {fieldname(line#)}. You can refer to a maximum of approximately 35 fields.
Exception: The computation produces a NULL result	
Is the value always equal to the computation?	Yes: field not stored No: field stored and recomputed if: <ul style="list-style-type: none">n You insert a new rown Its old value is nulln Its old value is not null, and any of a list of fields you specify is updated (even if updated to the same value)

- 2. At "Code to compute the result", enter the M code used to assign a value to the field.

3. At the "Exception:... If" fields, enter an M expression for each test which determines if the field should have a NULL value.
4. At the "Is the value always equal to the computation?" field, enter Yes or No.

If you enter No, you will see a pop-up auxiliary window at which you name the fields in this base table which, if updated, will cause this field to be computed:

-----Base Table Definition-----	
Base Table Name	Description
Guides_____	
Code to co (Explic SET {}=Dat	This field is computed:
	1. For a new row
	OR
	2. If its old value is null
Exceptio	OR
	3. If its old value is not null, and any of the following fields is updated:
I	(1/0)
I	
I	
Is the val	Exception: An explicitly specified value overrides the computation.
...Computed Field (Not Always)Unsaved Data Press <PF1><PF3> For Help	
Update Features	Conversion/ Validation Code
Error/Help Messages	Indexing Options
Copy Field	Additional Options

5. Add the names of the fields in this base table which, if updated, will cause this field to be recomputed.

Step 9: Define User Update Features

Select the Update Features horizontal menu option if you want to:

- n Increase the protection level above the default Add/Edit
- n Make a field required
- n Provide a default value for a field

Procedure To enter the Update Features option:

1. At the Field Definition master window, press <GO TO BOTTOM MENU>.

The highlight is on Update Features, which is the first horizontal menu option.

2. Press <RETURN>.

You will see the Update Features auxiliary window:

-----Base Table Definition-----+		
Base Table Name	Description	
Guides_____		
+-----Field Definition-----+		
Field Name	Description	
Title_____	Title of Guide_____	
+-----Update Features-----+		
Field Protection Add/Edit_____	Column Number 3	Field ID # 48
Required Field? No_____ If it is 'Maybe' Required -		
the M Expression to Determine if it is Required is:		
IF _____		
M Expression for Default (External) Value		

+-----+-----+-----+		
...Update Features Unsaved Data Press <PF1><PF3> For Help		
Conversion/ Validation Code		Error/Help Messages

Table 6-13 describes the fields on the Update Features window.

Table 6-13: Update Features Auxiliary Window

Field Name	Description
Field Protection	Defines level of user access to this field: AE Add/Edit A Add O Output only
Column Number	Open M with SQL generates the value in this field. It is an output only field; you cannot edit it. The Row ID is always column 1. In a child table, the Parent reference field is always column 0. Column number is used in the following ways: <ul style="list-style-type: none"> ▪ Determines the order in which fields are displayed when you issue an SQL SELECT statement. Field with subscript 0 listed first, and so on. ▪ The subscript to refer to this field when you use internal variables, such as %data and %edit, in SQL code.
Field ID #	Open M with SQL generates the value in this field. It is an output only field; you cannot edit it.
Required Field?	Y Yes N No M Maybe
If Maybe Required...	Required if "Maybe" is chosen for the Required Field? field.
M Expression for Default (External) Value	M code to define the default value to be displayed on a form or report that includes this field.

Defining Field Protection

You control the amount of access users have to field values when they are displayed on a Form Generator form. You specify one of the 3 levels of field protection which are described below from least protection to maximum protection:

- n **Add/Edit** Users may enter data into this field and later modify the data. This is the default.
- n **Add** Users may enter data into this field once, but may not modify it.
- n **Output** Values for this field are displayed on the window, but may not be modified. This protection level is appropriate for computed fields and other fields which will be displayed but not changed. On forms, the cursor will not go to fields with Output Only protection. An example of an output only field is the Column Number field on the Update Features window.

You Can Specify Stricter Protection in Form Generator

In the Form Generator, you can override this value, but only to a level of protection that is stricter. For instance, if you give a field Add protection in the Data Dictionary, you can only give it Output protection in the Form Generator. You can't decrease its protection by allowing Add/Edit access.

Example In the Guides table, we defined a field Latest_Edition_Gui as a designative reference to the Editions table. The value for this field is the row in the Editions table which is the most recent edition of the guide. This value is updated automatically via base table triggers, if necessary, when editions are inserted into the Editions table. Therefore, we must give this field Add/Edit protection.

We want to display the latest edition of each guide -- version and revision date -- in our display and reporting. Designative Reference fields cannot be displayed directly. However, if you use Form Generator forms, you create designative display fields from them. In the Form Generator, we create the Guides form, where we define two designative display fields based on Latest_Edition_Gui. One is the value of the Version field in Editions, the other is the value of the Revision_Date field. We give these designative display fields a protection level of output, since their value is determined by base table triggers.

Procedure To define field protection:

1. Enter the Field Definition master window.
2. Press <Go to Bottom Menu>.
3. Select Update Features.

Your cursor will be at the Field Protection prompt.

4. Press <RETURN> to select the default Add/Edit protection. Or press <Erase Field Line> and the first letter of the level of protection you desire (A or O).

Defining a Required Field

If you define a field as required, you cannot save a row of data entered in this base table until a valid value has been entered into the required field. On forms, the caption for a required field is displayed in reverse video, a color, or with underlining, depending on terminal type.

The Row ID is automatically defined as a required field.

It is recommended that you define as required those fields that users of the database will retrieve using an index map. See “Make Indexed Fields Required to Avoid Null Values” on page 9-3 in Chapter 9, *Index Maps* .

Required Only Upon Condition

You can specify that a field is required only if a certain condition is true. If you do this, you provide M code that specifies the condition.

Example In the Guides table, two fields — Date_Retired and Who_Retired — are only meaningful if the Guide has been retired. Therefore, we make those fields conditionally required, depending on the value of the field Retired. The completed Update Features window for the field Date_Retired is shown below:

+-----Base Table Definition-----+		
Base Table Name	Description	
Guides_____		
+-----Field Definition-----+		
Field Name	Description	
Date_Retired_____	Date Guide is Retired_____	
+-----Update Features-----+		
Field Protection Add/Edit_____	Column Number 3	Field ID # 49
Required Field? Maybe_____ If it is 'Maybe' Required - the M Expression to Determine if it is Required is: IF {Retired}="Y"_____		
M Expression for Default (External) Value _____		
+-----+		
...Update Features Unsaved Data Press <PF1><PF3> For Help		
Conversion/ Validation Code		Error/Help Messages

Procedure To define a required field:

- 1. Enter the Field Definition master window.
- 2. Press <Go to Bottom Menu>.
- 3. Select Update Features.
- 4. Move the cursor to the Required Field? option.
- 5. Answer “Y” for Yes, “N” for No or “M” for Maybe.

If you answer Maybe, you need to supply M code to define the condition when this field is required in Step 6.
- 6. Enter the M code that specifies the condition when this field is required.

Defining Default Field Values

The default value is the value that will be assigned automatically when inserting a new row. It will be saved with the row when the row is filed. It can be edited by the user if the field has Add/Edit protection.

The default value is used only for inserts, and only when an explicit value has not been assigned to the field. Default values are applied regardless of how the row is inserted: via a Form Generator form, an embedded SQL routine, or an SQL query.

Default Value for Row IDs on Forms

If you are using default physical structure, the M expression for default external value is defined as starting with number 1, then incrementing each Row ID value by 1. See “Open M with SQL Inserts Value in Row ID Field” on page 16-4 in Chapter 16, *Default Physical Structure* .

If you are using customized physical structure, and you define your Row ID to be one or more fields which contain actual values entered by users, you must ensure that the Row ID fields are not null before the rows are used on a Form Generator form.

One way to ensure that non-null values are set for these fields is to define default M expressions for them. Even when you expect the user to enter the value for the Row ID or a field it is based on, define a non-null default value as a dummy value, which the user can edit.

Restrictions on Default Values

The following restrictions apply to field default values:

- n They cannot contain references to other fields, using {fieldname} syntax.
- n They are not applied in an UPDATE situation, even if the field has a null value.

These limitations can be overcome by using the computed field feature in lieu of a default value. See “Step 8: Define Computed Field Calculations” on page 6-42.

Procedure To define the default field value:

1. Enter the Field Definition master window.
2. Press <Go to Bottom Menu>.
3. Select the Update Features option.
4. Move the cursor to the field M Expression for Default (External) Value.
5. Enter an M expression that specifies the default value.

Examples of Default Values

Example **Default Current Date:** To cause the current date to be displayed in a date field as the default value, enter the following M code at the Update Features window:

```
$ZDATE ( $H )
```

Example **Default for Yes/No Fields:** In the Guides table, we do not want to allow NULL values in the Yes/No field, Retired. Therefore, in addition to making the field required, we provide a default value of No:

-----Base Table Definition-----		
Base Table Name	Description	
Guides_____		
-----Field Definition-----		
Field Name	Description	
Retired_____	Is Guide Retired?_____	
-----Update Features-----		
Field Protection Add/Edit_____	Column Number 3	Field ID # 50
Required Field? Yes_____ If it is 'Maybe' Required -		
the M Expression to Determine if it is Required is:		
IF _____		
M Expression for Default (External) Value		
"No"_____		

...Update Features Unsaved Data Press <PF1><PF3> For Help		
Conversion/ Validation Code		Error/Help Messages

Step 10: Define Data Conversion and Validation Code

This is an optional step.

Select the Conversion/Validation Code option from the Field Definition window’s horizontal option menu. See Chapter 10, *Data Conversion and Validation* .

Step 11: Define Error and Help Messages

This is an optional step.

Select the Error/Help Messages option from the Field Definition window’s horizontal option menu. See Chapter 13, *Field Help and Error Messages* .

Step 12: Define Additional Options

This is an optional step.

Select the Conversion/Validation Code option from the Field Definition window’s horizontal option menu. From the Field Definition master window, press

<Go to Bottom Menu>. Select Additional Options from the horizontal options menu. Pressing Additional Options displays the following window:

-----Base Table Definition-----	
Base Table Name Guides_____	Description
-----Additional Options-----	
Field Name Retired_____	NOTE: Fields on an Index will Always be included in lookup lists
Data Type	Do not include in lookup lists? _____
Maximum Le	Don't use Template Information? _____
Number of	Report Column Title (0/1) Retired?_____
Multi-Line	----- Server Specific Options -----
Computed F	This field is a Date datatype in _____
...Additional Options Unsaved Data Press <PF1><PF3> For Help	
Update Features	Conversion/ Validation Code Error/Help Messages Indexing Options Copy Field Additional Options

The following table lists and describes the options located in the Additional Options window:

Table 6-14: Field Definition Additional Options

Do not include in lookup lists?	Specify Yes or No. No is the default. This option lets you exclude the current field from all lookup lists. <i>Yes</i> excludes this field from lookup lists. <i>No</i> allows this field to be included in lookup lists. See Chapter 8, <i>Base Table Lookups</i> .
Don't use Template Information?	Specify Yes or No. No is the default. This option lets you select whether or not to use the system-wide templates for field data types. For example, your System Manager may have established a template for all fields of data type Date so that, by default, they always display in format 3 with 4-digit years. <i>Yes</i> does not use these system-wide data type default templates. <i>No</i> uses the system's data type default templates.

Table 6-14: Field Definition Additional Options

Report Column Title	See “Define Column Titles for Reports” on page 6-54.
This field is a Date data type in	<p>This option applies only to applications that have been imported into the Open M with SQL Data Dictionary from non-Open M with SQL database systems, such as FileMan, MIIS, or other M systems. These applications use formats for dates and times that Open M with SQL cannot understand. Therefore, during import, Open M with SQL converts such fields to Text fields and stores the date/time data as text strings</p> <p>This option lets you associate foreign database fields with the name of an appropriate native format. Press <List Choices> to chose from a list of the following date/time formats:</p> <ul style="list-style-type: none"> » FileMan Date Format » FileMan Date/Time Format » M Time Stamp Format » MIIS Date Format » MIIS Date/Time Format <p>When you are using the Open M with SQL Relational Server, and a client application requests data from a field for which you have identified a foreign database native format, the Relational Server provides this information to the client.</p> <p>See Table 19-1: FileMan Preferences Options, on page 19-4 and Table 19-3: FileMan Data Types, on page 19-6 in Chapter 19, <i>The FileMan Interface</i>.</p>

Define Column Titles for Reports

You can define a title to be printed when a field is displayed as a data column in an M/PACT report. If you don't provide this title, M/PACT will use the value in the Description field.

You can override this value in M/PACT on a report-by-report basis. See the *Open M/SQL M/PACT Guide*.

Procedure To define a column title for this field in M/PACT reports:

1. Press <Go to Bottom Menu> at the Field Definition master window.
2. Move the highlight bar to the Additional Options option, and press <RETURN>.

You see the Additional Options window. The screen below shows a completed window for the field Retired in the Guides table:

-----Base Table Definition-----	
Base Table Name Guides_____	Description
-----Additional Options-----	
Field Name Retired_____	NOTE: Fields on an Index will Always be included in lookup lists
Data Type	Do not include in lookup lists? _____
Maximum Length	Don't use Template Information? _____
Number of	Report Column Title (0/1) Retired?_____
Multi-Line	----- Server Specific Options -----
Computed Field	This field is a Date datatype in _____
<div> ...Additional Options Unsaved Data Press <PF1><PF3> For Help </div>	
Update Features	Conversion/ Validation Code Error/Help Messages Indexing Options Copy Field Additional Options

- Enter as the Report Column Title the heading you want to appear on M/PACT reports for this field, then press <RETURN>. A report column title can consist of multiple lines, each line has a maximum of 45 characters.

For space considerations on reports, we provide a title shorter than the text we used in the Description field. If you do not define a report column title, M/PACT uses the Field Description (see “Steps 1 & 2: Specify a Field Name and Description” on page 6-4).

Step 13: Save the Field Definition

Once you have completed defining the fields in your base table, you need to save your definitions. As long as you don't reboot your computer or abnormally exit out of the Data Dictionary, your definitions will always be saved, or you will see a menu giving you a choice about saving your definition.

Procedure To save your field definition:

1. Press <PROCEED>.

Depending on the state of the PROCEED Save menu toggle, your field definition will be saved automatically or you see the PROCEED Save menu. See Chapter 3, *Using the Data Dictionary Interface*, for information on the PROCEED Save menu.

You return to the Field Definition row selection window.

2. To exit Field Definition, press <PREVIOUS>.

Copying a Field Definition

Rather than creating a field definition, you can copy the definition of an already defined field. You can:

- n Copy the current field within the same base table
- n Copy the current field to another existing base table
- n Copy a field into the current base table from the same base table
- n copy a field into the current base table from another base table

To copy a field definition, press <Go to Bottom Menu> from either the Field Definition main window or the Base Table Definition main window, and select Copy Field from the horizontal options menu. You can also perform a field copy from the Data Dictionary main menu by selecting the Copy a Base Table Field menu option.

Selecting Copy Field displays the following window:

-----Field Copy-----

Base Table to copy field from
Accounts_____

Field to copy
Client_Name_____

Base Table to copy field to

New field name

New field description

Field Copy
Press <PF1><PF3> For Help

Enter the name of the Base table to copy the field to.

Both base tables must be existing. You can press the <List Choices> key to see a list of existing base tables, or existing field names to copy from. Field Copy does not permit you to overwrite an existing field. Instead, it displays a message “There’s already a field in table: *x* with that name”

The following limitations apply to field copy operations:

- n You cannot copy Row ID or Parent Reference fields.
- n You cannot copy computed fields.
- n When copying fields across base tables, you cannot copy fields that reference other fields (using curly brace syntax: {field}) in the following locations:
 - External->Internal conversion code
 - Internal->External conversion code
 - Validation code
 - Additional Validation code
 - Default M expression

For further details on defining conversion and validation code, see Chapter 10, *Data Conversion and Validation* .

Editing and Deleting a Field Definition

If you have %ALTER privileges to a base table, you can change any part of its field definitions or delete field definitions, even after data has been entered into a base table.

Deleting a Field Definition

Be certain a field is not used in any views, reports or forms before you delete it. Once you delete a field, you cannot use SQL to reference data in that field.

Field is Deleted from Lookup

If you delete a field you specified in the Lookup Specification window, the field name at that window will be replaced with the message <<DELETED>>. If the field was used to lookup on, the lookup will no longer work. If the field was only a lookup display field, the lookup will still work.

Index Maps Deleted

If there is an index map for a deleted field, Open M with SQL deletes that index map.

Associated Global Data is Not Deleted

When you delete a field, the global that contains its data remains intact. You can still access it via M global reference. If you no longer need that data, you must write an application to delete the data.

Alternatively, you can write an SQL statement to delete the data values prior to deleting the field.

Procedure To edit or delete a field definition:

1. Enter the Data Dictionary.
2. Select the Base Table Definition option from the Data Dictionary menu.
3. Select the base table whose fields you wish to edit.
You see the Base Table Definition master window.
4. Move the cursor to <Fields> and press <RETURN>.
5. At the Field Definition row selection window, press <List Choices>.

The lookup box will display field name, description, and column number for each field in the base table:

-----Base Table Definition-----		
Base Table Name	Description	
Guides_____		
-----Field Definition-----		
Field Name	Description	
Date_Retired	Date Title Retired	3
Guides	Guides Row ID	1
Num_Editions	Number of Editions of Guide	4
Product	InterSystem Products	2
Retired	Is Title Retired?	5
Title	Name of Guide	6
Who_retired	Person Who Retired Title	7

...Field Definition Selecting Press <PF1><PF3> For Help		

- 6. Select the field you wish to edit or delete.
- 7. Edit the definition or press <Delete Row> to delete the field definition.

Implicit Joins

In many relational databases, the programmer must frequently join base tables with complex WHERE conditions in an SQL query. However, Open M with SQL provides two methods you can use to pre-define frequently occurring joins between base tables during base table definition: designative reference and characteristic relationship. These implicit joins simplify the writing of SQL queries for these tables and make data retrieval more efficient.

This chapter covers the following topics:

- n Use Designative Reference to Join Independent Tables page 7-2
- n Use Characteristic Relationship to Join Dependent Tables page 7-5
- n Benefits of Implicit Joins page 7-7

Use Designative Reference to Join Independent Tables

A designative reference is a join between tables in which one field in a row from the designating table contains the Row ID of a row in the designated table, thus providing access to all fields in the designated table.

Defining a Designative Reference Field

You define this field, giving it a data type of Designative Reference, when you define your base table. See “Defining a Designative Reference Field” on page 6-14 in Chapter 6, *Defining Base Table Fields*.

Tables Joined by Designative Reference are Independent

Unlike a characteristic (parent/child) relationship, a designative reference is a non-dependent link. Rows in the designated table exist independently of rows in the designating table. In other words, deleting a row from the designating table does not affect the row to which it points in the designated table.

Many-to-One Relationship

Rows in the designating table have a many-to-one relationship with rows in the designated table. This means that many rows in the designating table may point to the same row in the designated table.

Relational Foreign Key Implemented as a Pointer to Row ID

In relational database terminology, the designating table has a “foreign key” to the designated table. In M terminology, the designating table has a “pointer” to a Row ID of the designated table. In Open M with SQL, the value of a designative reference field in each row of the designating table is a Row ID of a row from the designated table.

Example of Designative Relationship Structure

In our sample application, the Staff_Ed table contains the field Name, which is a designative reference to the Docstaff table. This field allows the user to gain access to all fields in the Docstaff table when working with Staff_Ed in a form or SQL statement.

The table below shows the value of some rows of data from the Staff_Ed table. The Name field in the Staff_Ed table contains the Row ID of rows in the Docstaff table. Note the following:

- n Two rows in Staff_Ed point to the same row in Docstaff, indicating the many-to-one nature of designative reference.
- n The Staff_Ed table shows only a small subset of rows in that table.

Table 7-1: Staff_Ed

Name	Role
1	Chapters 2 and 4, peer review
1	Chapters 10 and 20
2	Chapter 1
4	Chapters 3,6,9,12

Table 7-2: Docstaff

Docstaff (Row ID)	Name	Phone	Street	City	State	Zip	Specialty
1	Kessler, Judy	111-2222	1 Elm St	Boston	MA	11111	UNIX, PC, Server
2	Ormiston, Land	222-3333	2 Main St	Boston	MA	11111	RDBMS
3	Page, Gail	333-4444	3 Oak St	Revere	MA	11211	M, VAX, M/NET
4	Russom, Phil	444-5555	4 Pine St	Boston	MA	11111	DataTree

Designative Reference is a One Way Outer Join

In Chapter 2, *The Open M with SQL Relational Database*, we discussed one-way outer joins. A designative reference is a type of one way outer join since it meets the following conditions:

- n The designative reference field may be empty; that is, it may not match a Row ID in the designated table.
- n If the designative reference field has a value, it must be a valid Row ID from the designated table.
- n Rows in the designated table may have no references in the designating table.

Multiple Designative Reference Fields

A table may have several designative reference fields, including one that designates itself.

Example of Designative Reference to Designating Table

Suppose you have a PATIENTS base table. You want to know if a relative of a patient is also a patient. You include a field which is a designative reference to the PATIENTS table, which may contain the Row ID of another row (patient) who is related to the current patient.

Designative Reference to a Child Table

In some cases, you may want to have a designative reference field to a row in a child table.

Example In the Guides table, the designative reference field Latest_Edition_Gui points to the row in the child table, Editions, which contains the most recent edition for a guide.

Designative References from Multiple Tables

A base table may be designated by any number of tables. For example, both the Editions and Staff_Ed tables in our sample application have a designative reference field to the Docstaff table.

Use Characteristic Relationship to Join Dependent Tables

A characteristic relationship is a link between base tables in which rows in one table (the child table) cannot exist independently of their corresponding row in the parent table. Therefore, if a row in a parent table is deleted, rows in the child table that are linked to that row are affected. In addition, Open M with SQL ensures that a child table can never be 'orphaned', i.e., exist without a parent table. If you attempt to delete a parent table that has children, you receive the message:

```
Must delete dependents first: childtable1, childtable2, ...,
childtablen
```

Automatically Created Trigger Deletes Child Rows

Open M with SQL provides a trigger that automatically deletes child rows if a parent row is deleted. You can, however, delete or edit that trigger. See Chapter 11, *Base Table Triggers*, for more information.

One-to-many Parent to Child Relationship

Parent rows have a one-to-many relationship with child rows. Each row in the parent table may point to zero, one or more than one row in the child table.

Many rows in the child table may be linked to the same row in the parent table. For this reason, a characteristic relationship, going from the child table to the parent table, can be thought of as a kind of designative reference.

Characteristic Relationship is an Inner Join

In a designative reference, rows in the designated table may contain no value in the designative reference field. Even though a characteristic relationship is a type of designative reference, going from child to parent, it is more restrictive since the join condition specifies that all rows of the child table **MUST** designate a parent table row. In this way, a characteristic relationship satisfies the relational definition of an inner join, which we discussed in Chapter 2, *The Open M with SQL Relational Database*.

Multi-Generation Parent-Child Relationships

Characteristic relationships are not limited to single parent/child relationships. A parent table may itself be a child of another table. Although a child table may only have one parent table, a parent table may have several child tables. For example, in our sample application, the Editions table has 2 child tables: Edition_Staff and Copy_Orders.

Child Tables Can Be Referenced Without Parent Table

Though characteristic relationships are hierarchical, child tables can be accessed using SQL without reference to the parent table, complying with the relational model.

Example of Characteristic Relationship

Using our sample application, the Editions table, shown in part below, is a child table of the Guides table (its parent), also shown in part. The existence of the Editions table is entirely dependent on its parent Guides table. If a row (a guide title) is deleted from the Guides table, its corresponding rows (editions) in the Editions table become “orphaned” and will be deleted via the automatic Open M with SQL trigger.

Note: Remember, you can edit this trigger to cause other actions to occur, such as transferring to an archive file.

Each row of the Guides table can have multiple editions, illustrating the one-to-many nature of a characteristic relationship. A Guides row might also have no rows in the Editions table, such as when a new title is agreed upon but before the first edition is published.

Table 7-3: Guides

Guides Row ID	Title	Latest Edition	Retired
1	ISM for UNIX System Manager's Guide		No
2	DT Windows		Yes
3	Open M with SQL Data Dictionary Guide		No

Table 7-4: Editions

Editions Row ID	Guides Parent Ref	childsub	Version	Revision_Date
Guides childsub	1	1	4.3	
	1	2	4.4	
	1	3	5.1	
	3	1	B	

Benefits of Implicit Joins

Designative references and characteristic relationships provide the following benefits:

- n Pre-define commonly used joins, allowing you to use Open M with SQL's simplified extended arrow syntax
- n Formally specify integrity constraints
- n Improve data access efficiency

Extended Arrow Syntax

When you define implicit joins, Open M with SQL provides a simplified form of SQL called extended arrow syntax to reference fields in the joined tables. You can use extended arrow syntax to write simplified SQL queries to represent:

- n Designative References
- n Child to Parent References in Characteristic Relationships
- n Parent to Child References in Characteristic Relationships

In extended arrow syntax, you use a dash followed by a greater-than symbol (->) to indicate a field in a joined table.

Extended arrow syntax causes an additional outer join condition to be added implicitly to the WHERE clause and the joined table to be added implicitly to the FROM clause of an SQL query.

Note: You *cannot* use double-arrow syntax (->>) to specify view joins. Syntax such as the following example is not permitted:

People->>Invoice->PrimaryDoc
Instead, use the following syntax for the same statement:
People.Invoice->PrimaryDoc

You can learn more about extended arrow syntax in the *Open M/SQL Developer Guide*.

Example of Extended Arrow Syntax for a Designative Reference

In our sample application, suppose you want to generate a list of the latest editions of each of the InterSystems Guides. Remember that the Guides table includes the field Title and the field Latest_Edition, which is a designative reference field to the table Editions and thus contains a Row ID from a row in the Editions table. In Editions, the name of the Row ID field is Editions. (All Row ID fields are named after their table in Open M with SQL.) Editions also contains the fields Version and Revision_Date.

If you do not have extended arrow syntax available, you need to write the following SQL query:

```
SELECT Guides.Title, Editions.Version, Editions.Revision_Date
FROM Guides, Editions
WHERE Guides.Latest_Version=*Editions.Editions
```

Using Open M with SQL's extended arrow syntax, you can write the following simplified query:

```
SELECT Title, Latest_Edition->Version, Latest_Edition->
Revision_Date
FROM Guides
```

Referential Integrity Constraints

Implicit join definitions include built-in integrity constraints. For instance, the existence-dependent relationship between the `Edition_Staff` table and the `Editions` table may be regarded as an integrity constraint: no edition staff can exist without a corresponding edition.

Efficient Data Access

Built-in pointers speed data retrieval.

Base Table Lookups

If your application uses Form Generator single-row forms for data entry or display, you need to define lookups for form users. Lookups are used by single-row forms to allow users to select a row from the base table which is the data source for the form. You can define lookups at the base table level or at the form level.

This chapter describes the different types of lookups and how to define them. Specifically, it covers the following topics:

n What is a Lookup?	page 8-1
n Where to Define Lookups	page 8-3
n Lookup Specifications	page 8-3
n Types of Lookup Queries	page 8-5
n Defining Base Table Lookups	page 8-7
n Defining Lookups for a Child Table	page 8-13
n Matching	page 8-14
n Customizing a Lookup Query	page 8-18

What is a Lookup?

In an Open M with SQL application, users typically populate base tables using forms created with the Open M with SQL Form Generator. A database form is always associated with one base table and allows users to add, edit, and delete information from that base table. Forms can be multi-row or single-row. Multi-row forms work with multiple rows of data simultaneously. Single-row forms work with one row of data at a time. The first step in using a single-row form is to select a row. Single-row forms use lookups to select rows.

When a user runs a single-row form, the first window to appear is the row selection window, which contains a group of **lookup fields**. These lookup fields are the mechanism through which a user can identify and retrieve a row from the

n **No match:** Open M with SQL asks the user if this is a new entry.

n **One match:** You can choose between the following two modes of behavior:

- Open M with SQL displays a lookup box containing the one match
- Open M with SQL retrieves the matching row directly into the form

n **Many matches:** Open M with SQL displays a lookup box containing a list of all matches.

Example This example illustrates the row selection process for a single-row form.

A user enters the letter "D" in the Title field and presses the <Search Current Table> key to initiate the lookup query. Open M with SQL uses the lookup query based on the Title field alone, finds the rows where "Title" begins with the letter "D" and displays these matches in a lookup box. The fields displayed in the lookup box include the Title of the Edition, the version of the product it documents, the revision date of the document and the date the order was placed.

[illegible]

Where to Define Lookups

You can define lookups at two levels:

1. Base Table

You may define lookups associated with a base table in the Data Dictionary, where they serve as the default lookups for all forms associated with that base table.

2. Form

You can also define lookups for any single-row form in the Form Generator, where they override any default lookups defined for the associated base table. If you do not define default lookups for a base table, you must define lookups for every single-row form associated with that base table.

You only need to define lookups if you plan to use forms.

Lookups and Index Maps

There is an advantage to defining lookups at the base table level. When base tables employ default physical structure, Open M with SQL automatically generates an index map for each lookup field. If you have a large database, the speed and efficiency of lookup performance may significantly improve if the lookup fields are indexed.

If you define override lookup specifications at the form level, you should make sure that all lookup fields are indexed in the originating base table.

If you define your own maps, you should make sure to define indexes for all lookup fields.

Lookup Specifications

To define lookups, you must provide certain basic information known as **lookup specifications**. Lookup specifications consist of two elements, **lookup fields** and **lookup display fields**.

Lookup Fields

Lookup fields are the fields that appear on the row selection window of a single-row form. Lookup queries use these fields to select rows from the database by matching user input to actual database values.

If you specify no lookup fields in a lookup query, you create an unqualified lookup query. This allows users to see a complete list of all rows in the database

by simply pressing the <Search Current Table> key from an empty row selection window.

If you specify more than one field per lookup query, the user must supply input for each of the lookup fields before Open M with SQL can execute the lookup query.

A single lookup query should contain no more than 6 lookup fields and 6 lookup display fields. From this number you should subtract one field for each level of dependency of the table. For example, 5 lookup fields maximum for a child table, 4 lookup fields maximum for a grandchild table, etc.

Lookup Display Fields

Lookup display fields are the fields whose values are displayed in the lookup box for each row returned by the lookup query. These fields do not impact row selection. Their purpose is simply to aid the user in accurately identifying the row to select.

You may want to display the same fields as those used to look up on. Or, you may want to display different fields altogether. The choice is yours, except that you may not select multi-line fields as lookup display fields.

You should limit the number of lookup display fields, so that the field column titles and values don't overextend the width of the screen, thus causing the row lookup display to be randomly truncated. A maximum of 6 lookup display fields is suggested, because a lookup box can accommodate a maximum of 6 lookup display fields.

Types of Lookup Queries

From the lookup specifications defined by the programmer, Open M with SQL generates lookup queries. A **lookup query** is the vehicle for searching the database and matching user input to actual values. You may provide lookup specifications for one, two, or many lookup queries for a single base table.

When a user enters information into the lookup fields on a row selection window and presses the <Search Current Table> key, Open M with SQL invokes the appropriate lookup query. The lookup query searches the database for rows which match the lookup information and returns a list of those rows in a lookup box.

You can define three different types of lookup queries: **unqualified lookup queries**, **single field lookup queries**, and **compound lookup queries**.

Unqualified Lookup Query

An unqualified lookup query has no lookup fields. If defined, the unqualified lookup query is invoked when a user presses the <Search Current Table> key from a blank row selection window (no lookup information is provided). The unqualified lookup query returns a lookup box that lists all rows in the associated base table. This constitutes an unrestricted, global lookup. An unqualified lookup can be quite useful for small tables but impractical for very large ones.

If you do not define an unqualified lookup query, the user will not be able to perform a global lookup on the associated base table.

Observe These Restrictions When Defining An Unqualified Lookup Query

1. You can define only one unqualified lookup query per base table.
2. An unqualified lookup query cannot exist by itself as the only lookup query defined for a base table. It must be accompanied by at least one additional lookup query that contains at least one lookup field.

Single Field Lookup Query

A single field lookup query has one lookup field. When a user enters information into the specified lookup field in the row selection window and presses the <Search Current Table> key, Open M with SQL invokes the single field lookup query for that field. The display fields defined for the lookup query appear in a lookup box that lists all rows in the associated base table whose actual values match the lookup information.

You can define many single field lookup queries.

Compound Lookup Query

A compound lookup query has two or more lookup fields. Compound lookup queries simultaneously match lookup information in multiple lookup fields to those same fields in base table rows.

When a user enters information into each of the specified lookup fields in the row selection window and presses the <Search Current Table> key, Open M with SQL invokes the compound lookup query. The display fields defined for the compound query appear in a lookup box that lists all rows in the associated base table whose actual values match the lookup information.

You can define multiple variations of compound lookup queries per base table.

Enabling Lookup Queries

You can define all three types of lookup queries within the lookup specifications for any given base table, or even multiple variations of the single field and compound lookup queries. To enable a lookup query type, you must define the requisite lookup specifications. To invoke a lookup query, the user must specify lookup information in the appropriate fashion. Table 8-1 lists the three lookup query types and shows what the programmer must do to set them up and what the user must do to initiate them.

Table 8-1: Lookup Query Types

Lookup Query Type	To Define	To Invoke
Unqualified	No lookup fields are specified.	The user enters no lookup information, then presses the <Search Current Table> key.
Single Field	Exactly one lookup field is specified.	The user enters lookup information into that particular field only, then presses the <Search Current Table> key.
Compound	Two or more lookup fields are specified.	The user enters data into each lookup field, then presses the <Search Current Table> key.

Precedence Given to Most Restrictive Queries

When multiple lookup queries exist simultaneously for a single base table, Open M with SQL tests the list of lookup queries in order of most restrictive to least restrictive, and invokes the first lookup query to test positive, regardless of whether or not it yields a successful database match.

The restrictiveness of a lookup query is determined by how many lookup fields it has — the more lookup fields it has, the more restrictive the query. For instance,

if you have three lookup queries with 0, 1, and 2 lookup fields respectively, the 2-field lookup query (compound) is the most restrictive and thus will be tested first, followed by the 1-field query (single field), and finally the 0-field query (unqualified). Likewise, when multiple compound lookup queries exist, Open M with SQL always tests the most restrictive of them first.

Example The example below illustrates the principles behind lookup query precedence.

If you define three different lookup queries with the following lookup fields:

Lookup Query #1	Lookup Query #2	Lookup Query #3
Lookup Fields:	Lookup Fields:	Lookup Fields:
Name	Name	Name
Organization	State	Organization
		State

Open M with SQL tests first for Lookup Query #3. If the user enters information into all three lookup fields, Open M with SQL always invokes Lookup Query #3.

If the user leaves one of the lookup fields blank, Open M with SQL first tests Lookup Query #3, but cannot invoke it. It then tests the two-field lookup queries in the order in which they were defined, and invokes the first one whose conditions are met by the user input.

Defining Base Table Lookups

This section describes how to define base table lookups for a base table that is not a child table. A slightly different procedure is followed to define base table lookups for a child table, as described in “Defining Lookups for a Child Table” on page 8-13.

Lookups defined at the base table level are the default lookups for every single-row form associated with that base table.

Base table lookups are optional; it is not required that you define them. However, if you do not define them, you must define lookups in each Form Generator single-row form which is based on the base table.

To validate base table lookups, you must place all lookup fields on the master window of each associated form.

Procedure To define lookups for a base table:

1. Enter the Base Table Definition master window.

- You see the Row Selection Lookup Specifications window, as shown below:

```

+-----Row Selection Lookup Specifications-----+
|
|  Lookup Field          Display Fields          Query
|  _____          _____          Options
|  _____          <..>          <Options>
|
|  _____          <..>          <Options>
|  _____
|
|  _____          <..>          <Options>
|  _____
|
|  _____          <..>          <Options>
|  _____
|
+-----+

```

Row Selection Lookup Specifi
Press <PF1><PF3> For Help

Table 8-2 lists and describes the fields located on the Row Selection Lookup

Table 8-2: Fields on Row Selection Lookup Specifications Window

--	--

Field Name	Description
Lookup Field	This field provides several lines where you list the fields used in the lookup query to search the database. List one field per line. These fields appear on the row selection window of single-row forms. You can specify any displayable single-line base table field, including the Row ID (if it is defined to be displayable). Multi-line fields and Designative Reference fields are not allowed. You should observe an upper limit of six lookup fields (fewer for child tables).

Table 8-2: Fields on Row Selection Lookup Specifications Window

<...> (Display Fields)	Press <RETURN> on this branching field to invoke the Display Field Definition window for the row that appears in the lookup box. There you can specify one or more display fields, the header associated with each field, and the display length of the header. You can specify any displayable single-line base table field, including the Row ID (if it is defined to be displayable). Multi-line fields and Designative Reference fields are not allowed. Lookup display fields can reiterate lookup fields. You should observe an upper limit of six lookup display fields.
<Options>	Press <RETURN> on this branching field to invoke the Query Options window, where you can modify certain attributes of the generated lookup query at the SQL level. See "Customizing a Lookup Query" on page 8-18.

3. In the Row Selection Lookup Specifications window, enter lookup fields in the Lookup Fields column.
- You may press the <List Choices> key to see a popup window that lists all fields defined for the base table.
- To define an unqualified lookup query, leave the Lookup Fields column empty.
4. Optionally, press <RETURN> on the Display Fields <...> branching field to see the Display Fields Definition window, as shown below:

-----Row Selection Lookup Specifications-----

Lookup Field

>> Date_Retired

Display Field

Date_Retired (8)

Header: Dates (5) 15

Header: ()

Header: (0)

Header: ()

Header: ()

Header: ()

Display Length

-----Row Selection Display Demo-----

Dates.....

Display Fields

Press <PF1><PF3> For Help

Saved...

Enter lookup display fields in the Display Field column. This must be the name of an existing field in the base table. You may press the <List Choices>

key to see a lookup box that lists all fields defined for the base table. You must enter at least one field.

The maximum length defined for that field is shown to the right of the field name; in the above example, it is an 8-character date field.

You may also define a header text for this display field. A header text may be up to 75 characters long (see below), and contain any characters. The length of the header text is shown to the right of the header text; in the above example “dates” is 5 characters long.

You may specify a Display Length for each header text. If the length you specify is larger than the header text length, it pads the text with blank spaces; if you length you specify is smaller than the header text length, it truncates the text. In the above example, the header’s display length is 15 characters long. As shown in the Row Selection Display Demo window at the bottom of the previous screen, the header text is padded with blank spaces to the length specified in the Display Length field. If you specify more than one header, these headers and their spacing are shown in the Row Selection Display Demo window.

You cannot specify a total header display length of more than 75 characters. If you specify more than one header, the system does not permit you to define header display lengths that total more than 75 characters. When totaling header display lengths, remember to include the 2 blank spaces that M with SQL automatically inserts to separate these headers.

5. You may continue to define lookup specifications for as many lookup queries as you wish.
6. When you are finished defining lookup specifications, press <PROCEED> to save your definitions and exit the Row Selection Lookup Specifications window back to the Base Table Definition master window.

Designative Display Fields in Lookup Specifications

Lookup fields and lookup display fields can be any displayable single-line fields defined for the base table, including the Row ID (if it is defined to be displayable). Multi-line fields and Designative Reference fields are not allowed. Designative display fields, since they are displayable, are allowed.

A designative display field is a form-only field created from a Designative Reference field that retrieves information from a specified column in the designated table.

You can spontaneously create designative display fields in the Row Selection Lookup Specifications window of a base table definition for the purpose of including them in your lookup specifications. This applies both to Lookup Fields and to their Display Fields.

Open M with SQL makes any designative display fields created in this fashion available to all forms associated with the base table.

For more information about designative display fields, see the *Open M/SQL Form Generator Guide*.

Procedure To create designative display fields in a base table lookup specification:

1. From the Base Table Definition master window, press <RETURN> on the <M/WINDOWS Lookups> branching field to access the Row Selection Lookup Specifications window.
2. In the Lookup Fields (or Display Fields) column, enter the name of a Designative Reference field currently defined in the base table.
3. Append the extended arrow syntax (->) to the name of this Designative Reference field.

For information about extended arrow syntax, see Chapter 7, *Implicit Joins*.

4. Press <RETURN>.

You see a lookup box that lists all possible designative display fields that can be created from the specified Designative Reference field. This is the list of all fields defined for the designated table:

Row Selection Lookup Specifications		
Lookup Field	Display Fields	Query Options
	<..>	<Options>
	<..>	<Options>
-----more-----		
Main_Author	Des Ref to Author Table	
Main_Author->Date_Retired	Date Title Retired	
Main_Author->Guides	Guides Row ID	
Main_Author->Num_Editions	Number of Editions of Guide	
Main_Author->Product	InterSystem Products	
Main_Author->Retired	Is Title Retired?	
Main_Author->Title	Name of Guide	
Main_Author->Who_retired	Person Who Retired Title	

Row Selection Lookup Specifi
 Press <PF1><PF3> For Help

5. Use the <Up Arrow> and <Down Arrow> keys to move the highlight bar to the appropriate designative display field in the lookup box, and press <RETURN> to select it.

In this way, you may define as many designative display fields as you wish in your lookup specifications.

Remember that all lookup fields in a lookup query must be present on the master window of a form to make the lookup query valid. This means that any designative display fields defined as lookup fields (not lookup display fields) must be present on the master window of all associated forms to make the lookup query valid.

Example The following window shows the lookup specifications for three lookup queries defined within the Copy_Orders base table for the "Copy Orders" form (the Display Field entries for all three lookup queries are the same):

-----Row Selection Lookup Specifications-----		
Lookup Field	Display Field	Display Length
>> _____	Title _____ ()	
	Header: _____ (0)	---
Title _____	Version _____ ()	
	Header: _____ ()	---
Title _____	Revision_Date _____ ()	
Version _____	Header: _____ ()	---
	Order_Date _____ ()	
	Header: _____ ()	---

-----Row Selection Display Demo-----

NO ROW SELECTION HEADER DEFINED

Display Fields Press <PF1><PF3> For Help

Saved...

Lookup Query 1 - unqualified lookup query

Open M with SQL invokes this query when the user presses the <Search Current Table> key from a blank row selection window. It returns all rows in the associated base table. As deifined here in the Display Field box, the lookup box will display the "Title", "Version", "Revision_Date", and "Order_Date" fields.

Lookup Query 2 - single field lookup query

The field "Title" is the only lookup field defined. Open M with SQL invokes this query when the user types lookup information into the "Title" field on the row selection window and presses the <Search Current Table> key. The lookup box will display the "Title", "Version", "Revision_Date", and "Order_Date" fields for all base table rows whose "Title" field matches the user input.

Lookup Query 3 - compound lookup query

Open M with SQL invokes this query when a user types lookup information into both the "Title" and "Version" fields on the row selection window and presses the <Search Current Table> key. The lookup box will display the "Title", "Version",

"Revision_Date", and "Order_Date" fields for all base table rows whose "Title" and "Version" fields match the user input.

Defining Lookups for a Child Table

A **child table** is a base table that is existence-dependent upon another base table, its parent. The term **characteristic relationship** denotes the relationship of a child to its parent. In a characteristic relationship, all rows in a child table are existence-dependent on a single row in the parent table. For example, in the characteristic relationship between the parent table "Guides" and child table "Editions", every row in the Editions table must correspond to one and only one row in the Guides table, and a row in the Guides table may have many (or no) corresponding rows in the Editions table.

Two Types of Lookups

You can perform a lookup on rows in a child table either with reference to its parent or without reference to its parent. For example, you can look up on editions for a specific Guide (referenced parent), or you can look up on editions for all guides (unreferenced parent). When the parent reference is "known", the lookup is restrictive. When the parent reference is "not known", the lookup is unrestricted.

Open M with SQL lets you define default Data Dictionary lookups for a child table in any of the following three ways:

- n One set of lookups where parent is known

You may only create a form for this child table that is called from a single-row form based on the parent table, so the parent Row ID is known.

- n One set of lookups where parent is not known

You may create a stand-alone form for the child table.

- n Two sets of lookups: one with parent known, one with parent unknown

Defining two sets of lookups, one where the parent is "known" and another where the parent is "not known", allows you to create both types of forms for the child table.

Procedure To define lookups for a child table:

1. Enter the Base Table Definition master window of a child table.
2. From the Base Table Definition master window, press <RETURN> on the <M/WINDOWS Lookups> branching field.

You see the Lookups for Characteristic Tables window, as shown below:

-----Base Table Definition-----

Base Table Name

Description

Editions

Use Default Physical Structure? Yes_

< Fields >

< M/WINDOWS Lookups >

< Physical St

Approximate N

+Lookups For Characteristic Tables+

< Lookups - parent is known >

< Lookups - parent is not known >

Implicit Validation BaseTable Compile Comp Rel Copy Advanced Comments

...Lookups for Characteristic Tables

Press <PF1><PF3> For Help

The Lookups for Characteristic Tables window lets you define lookup specifications for either case of parent reference availability ("known" or "not known").

- 3. Select the <Lookups - parent is known> or the <Lookups - parent is not known> action field.

In either case, you see the Row Selection Lookup Specifications window, as shown on page 8-8.

Continue to define lookup specifications as described in steps 3-6 of the procedure for defining base table lookups.

Matching

When a user presses the <Search Current Table> key to initiate a database search and Open M with SQL invokes the appropriate lookup query, **matching** is the action performed to retrieve rows from the database. The lookup query compares information entered by the user into lookup fields to actual database values. When values in one or more base table rows correspond to the user input, matching is successful. If twelve database rows correspond, there are twelve matching entries or "matches". When no database rows correspond to the lookup information, matching is unsuccessful.

The following conditions must be met for matching to be successful for a particular lookup query:

- n All lookup fields specified in the lookup query must be present on the master window of the form.
- n The user must enter lookup information into all lookup fields specified in the lookup query.
- n The lookup information entered by the user must match at least one actual database row.

Open M with SQL provides a number of functions to make single row selection more robust. These functions transform user input and/or actual database values in order to minimize or eliminate superficial differences between them.

- n **Field Matching functions** allow partial entry of lookup fields and eliminate unwanted case sensitivity.
- n **Field Conversion Code** converts external representations into internal values, particularly for Time, Date, and Number fields where one value may have many different external representations.

Field matching applies only to qualified lookup queries, i.e. lookup queries that have lookup fields. The default matching features of each lookup field in a qualified lookup query are determined by the field's Data Type, as defined in the Base Table Definition master window.

Field Matching Functions

Partial or Exact Matching

Lookup fields with Data Type Name, Text, and Multiple Choice automatically support **partial matching**. Partial matching means that the lookup information entered by the user need only match a "starts with" portion of the field's actual value. Any sequence of characters entered as lookup information is sufficient to qualify a partial match. With partial matching, the input "Smith,J" would match the actual values "Smith,John", "Smith,John C", and "Smith,Jennifer". Or, the input "Thomas" would match the actual values "Thomas,Gary" and "Thomasson,Anne". Partial matching is implemented by means of the %STARTSWITH comparison operator, described in "Lookup Queries Use InterSystems' SQL Extensions" on page 8-21.

The opposite of partial matching is **exact matching**. Exact matching means that the lookup information entered by the user must exactly replicate the field's actual database value. Exact matching always identifies one unique row, or fails. For example, suppose a lookup query with a lookup field of Data Type Name. If the user enters "Smith, John" and the closest actual database value is "Smith, John C", an exact match would fail. You can enable exact matching for fields of Data Type Name, Text, and Multiple Choice by removing the %STARTSWITH

Case Transformation

When you define a field with a data type of Name or Text, you see the following window:

[illegible]

n	EXACT	no transformation of the actual values (default).
n	UPPER	converts lowercase letters to UPPERCASE.
n	ALPHAUP	strips all spaces and punctuation, then runs UPPER.

If a field of Data Type Name using the ALPHAUP case transformation function has the actual database value "O'Reilly, Susan Jean", it permits successful matching on any of the following lookup information entries:

oreilly,susanjean

OREILLY,Susan-Jean

O'REILLY,SUSAN JEAN

OREILLY,SUSANJEAN

If this field were defined as a Text field, the converted value would be:

OREILLYSUSANJEAN

Using ALPHAUP or UPPER case transformation, you can maximize the scope of matches yielded by a lookup query. EXACT case conversion results in an unforgiving comparison test, in which an errant space or incorrect capitalization can disqualify an otherwise exact match.

The ALPHAUP function is often the best choice for Name and Text fields that are used in lookups. ALPHAUP strips punctuation and converts all letters to uppercase. Thus, lookups on Name and Text fields are insensitive to case and punctuation; the user can type all or part of an entry into a lookup field in upper or lower case, with or without punctuation, and Open M with SQL will find the matching rows.

Note: Open M with SQL properly handles non-English characters; thus, for example, "ü" (Umlaut in German) is not stripped as punctuation by ALPHAUP.

Case Transformation and Ordering

When the case transformation function acts on a field that is named in the ORDER BY clause, it can affect the ordering sequence for rows retrieved into the lookup box. For more information, see the section of this chapter entitled "Lookup Queries Use InterSystems' SQL Extensions" on page 8-21.

Field Conversion Code

Whereas field matching functions act on the external representation of a field, field conversion code affects the internally stored value of a field.

Open M with SQL automatically generates conversion code for fields of Data Type Date, Time, and Number. You do not need any case conversion to perform lookups on these fields because Open M with SQL stores dates, times, and numbers in a unique internal format, where same values are always identical regardless of the external format.

For example, a Date field with an external value of "3-10-91" or "3/10/91" is converted to 54886, an internal representation of this date. This internal storage value is used in matching. Therefore, the SQL lookup query generated to look up values by a date need not contain any case conversion functions.

Unlike field matching functions, which are reflected in the SQL lookup queries generated to perform row selection, conversion code is written in M code. You can edit this conversion code for a database field within its Field Definition window, as described in Chapter 10, *Data Conversion and Validation*.

Note: Fields of Data Type Multiple Choice are a special case, where automatic generation of conversion code is optional. If you spec-

ify both external and internal values, Open M with SQL default-generates the appropriate conversion code. If you do not specify internal values, no conversion code is generated.

Summary of Case Transformation and Field Conversion Code

The following table summarizes the default case transformation functions and the behavior of field conversion code for all Data Dictionary Data Types. The Index Value column describes the value stored in the default Index Map Open M with SQL creates for all fields defined as Lookup Fields.

Table 8-3: Field Conversion Code and Case Transformation Functions

Data Type	Default Case Function	Field Convert Code	External Value	Internal Value	Index Value
Date	N/A	Automatic	3/10/91	54886	54886
Designative Reference	N/A	N/A	45004	45004	45004
Multiple Choice	N/A	Optional	RED	R	R
Name	EXACT	Custom	Paleo,John	Paleo,John	Paleo,John
	UPPER		Paleo,John	Paleo,John	PALEO,JOHN
	ALPHAUP		Paleo,John	Paleo,John	PALEOJOHN
Number	N/A	Automatic	34.300	34.3	34.3
Row ID	N/A	N/A	12	12	12
Text	EXACT	Custom	Paleo,John	Paleo,John	Paleo,John
	UPPER		Paleo,John	Paleo,John	PALEO,JOHN
	ALPHAUP		Paleo,John	Paleo,John	PALEOJOHN
Time	N/A	Automatic	3:09PM	54557	54557
Yes/No	N/A	Automatic	Yes	Y	Y

Customizing a Lookup Query

For each set of lookup specifications defined by the programmer, Open M with SQL generates a default SQL lookup query. You can optionally modify certain attributes of the generated SQL code to create an override SQL lookup query. This allows you to exercise precise control over lookup behavior.

To customize a lookup query, you can modify the WHERE and/or ORDER BY clauses of the SQL-generated query. This enables you to alter its behavior in the following ways:

- n Change a partial match lookup query to an exact match, or vice-versa (by changing the comparison operator).
- n Change the case transformation function.
- n Make a lookup query more restrictive by excluding rows that do not meet some set of criteria.
- n Modify the order of rows displayed in the lookup box.

You Can Toggle Between Default and Override Versions

Open M with SQL can maintain two versions of the SQL-generated lookup query, the default version and an override version. You can toggle between the two versions by way of the Use Default Query field in the Lookup Query Definition window.

The version last selected before saving and exiting the Lookup Query Definition window is the de facto version.

Suppress Lookup Box

You can also modify a lookup query to suppress the appearance of the lookup box when the query returns one unique match. When the lookup box is suppressed, Open M with SQL invokes the form directly with the unique match.

When presented, the lookup box forces the user to perform the additional step of pressing <RETURN> to confirm the selection, but it also gives the user the option of refusing the match if it was not intended.

By default, every lookup query presents the lookup box.

Procedure To customize the default SQL-generated lookup query:

1. Enter the Row Selection Lookup Specifications window.
2. Define lookup specifications for at least one lookup query.
3. Press <RETURN> on the < Options > branching field located to the right of the lookup specifications for a particular lookup query.

You see the Query Options window, as shown below:

-----Row Selection Lookup Specifications-----

Lookup Fi

>> Name

-----Query Options-----

Suppress lookup window for unique match? ____

Use Default Query? Yes_ Support Searched Lookups? ____

-----Default Query-----

DISTINCT or DISTINCT BY (...): _____

SELECT people,Name,City,State

FROM People

WHERE / GROUP BY / HAVING / ORDER BY:

WHERE (Name %STARTSWITH ::Name) _____

ORDER BY Name _____

< Searched ORDER BY: >

< Lookup Box Override Specs > < Additional Info >

Query OptionsPress <PF1><PF3> For Help

Table 8-4 lists and describes the fields located on the Query Options window:

Table 8-4: Fields on Query Options window

Field Name	Description
Suppress lookup window for unique match?	This is a Yes/No field. It tells Open M with SQL whether or not to present a lookup box when the lookup query yields exactly one unique match. Answer <i>Yes</i> to suppress the appearance of the lookup box, causing the query to directly invoke the form. Answer <i>No</i> to present the lookup box. <i>No</i> is the default response.
Use Default Query?	This is a Yes/No field. It lets you toggle between the default version and override version of the lookup query, as displayed in the bottom half of this window. Changing this value changes the text of the divider in the middle of this window. Answer <i>Yes</i> if you want to use the default lookup query. Answer <i>No</i> if you want to edit a copy of the default lookup query and create an override lookup query. <i>Yes</i> is the default response.
Support Searched Lookups?	This is a Yes/No field. Answer <i>Yes</i> to allow lookup on the first field displayed in the lookup list. Additional characters typed will narrow selection. <i>Yes</i> is the default response.
DISTINCT or DISTINCT BY (...)	The field allows you to restrict the override lookup query to allow DISTINCT values only. If you enter a DISTINCT or DISTINCT BY clause, Open M with SQL enters this clause after the SELECT keyword and before the first SELECT list item of the query.
<Searched ORDER BY: >	A read-only field. Only activated if you specified Support Searched Lookups=YES, it displays the search order variable. For example, %UPPER().

Table 8-4: Fields on Query Options window

<Lookup Box Override Specs>	This action field displays a window that permits you to define the the coordinates of a lookup box: the top left 'x' coordinate, the top left 'y' coordinate, and the lookup box height. If there is not enough space for the coordinates you specify, Open M with SQL determines the best fit.
<Additional Info>	This action field displays a window showing the date and time of the most recent edits to this query.

4. Answer *Yes* or *No* (as preferred) to the Suppress Lookup Box question, and press <RETURN>.
5. Answer *No* to the Use Default Query? prompt, and press <RETURN>.

This replaces the default lookup query with an override lookup query and changes the title from "Default" to "Override". You can now access the SQL code inside the lookup query. Your cursor begins on the WHERE clause.

Note: You can later recall the default lookup query by simply changing the value in the Use Default Query field to **Yes**.

6. Modify the WHERE and/or ORDER BY clauses of the override lookup query to your desired specifications.

You must use valid SQL syntax. See below for hints on editing the SQL code.
7. When you have finished editing the override lookup query, press <PROCEED> to save your edits and exit the Lookup Query Definition window back to the Row Selection Lookup Specifications window.
8. Press <PROCEED> again to exit the Row Selection Lookup Specifications window back to the Base Table Definition window.

Lookup Queries Use InterSystems' SQL Extensions

In order to customize an SQL-generated lookup query, you need to understand the SQL conventions used to build these queries. These conventions include several InterSystems' SQL extensions, which are described below. For more information on InterSystems' SQL extensions, refer to the *Open M/SQL Developer Guide*.

%STARTSWITH

The keyword %STARTSWITH is the agent of partial matching. An InterSystems' SQL extension, %STARTSWITH is a comparison operator that tests whether one

character string is a prefix of another. The syntax of this comparison is as follows:

```
WHERE expr1 %STARTSWITH expr2
```

where "expr" is any valid SQL expression.

For example, a query containing the clause:

```
WHERE City %STARTSWITH 'S'
```

returns all rows whose "City" field starts with the character "S".

Fields of Data Type Name are treated differently by %STARTSWITH. If the field "Name" has a Data Type of Name, a query containing the clause:

```
WHERE Name %STARTSWITH 'SM,J'
```

returns all rows for which \$PIECE(Name,"",1) starts with "SM" and \$PIECE(Name,"",2) starts with "J", for example:

```
Name = "SMITH,JANE"  
Name = "SMOOT,JERRY"
```

but not:

```
Name = "SMITH,HERMAN"
```

%STARTSWITH is the default-enabled comparison operator for fields of Data Type Name, Text, and Multiple Choice. This means partial matching is always the default lookup query behavior for fields of these types.

To disable partial matching, replace %STARTSWITH with another SQL comparison operator ("=" for exact matching).

Case Transformation Functions

The Open M with SQL case transformation functions (ALPHAUP, UPPER, and EXACT) are implemented in SQL as %ALPHAUP, %UPPER, and %EXACT. Open M with SQL default-generates these keywords into lookup queries with lookup fields of Data Type Name and Text. This determines the case transformation syntax for user input. You may change the case transformation syntax for any lookup field by changing the SQL case transformation keyword. EXACT is the default transformation syntax, and thus does not need to be explicitly specified. For further details refer to the *Open M/SQL Developer Guide*.

The transformation function used by a field affects the results of an ORDER BY clause involving that field. It also affects comparisons on that field using any of the following comparison operators:

>
 <
 =
 >=
 <=
 NOT>
 NOT<
 %STARTSWITH

User Input Variables

The SQL-generated lookup query uses special variables to store lookup information entered by the user into a lookup field on a row selection window. These variables can be represented in either of two equivalent ways:

- n By placing two colons in front of the lookup field. For example, if the lookup field is "Name", the corresponding user input variable would be "::Name".
- n By enclosing the lookup field name in curly braces and placing a single colon in front of it. For example, if the lookup field is "Name", the corresponding input variable would be :{Name}

SQL uses the following syntax to represent the comparison of lookup information entered by a user with the actual database values of a lookup field:

WHERE field_name comparison_op input_variable

For example:

WHERE Name %STARTSWITH ::Name

or

WHERE Name = :{Name}

Example Override Lookup Query

If you enter the following set of lookup specifications for the "Docstaff" base table:

<u>Lookup</u>	<u>Display</u>
<u>Fields</u>	<u>Fields</u>
Name	Name
City	City
	State

Open M with SQL default-generates the following SQL lookup query:

```
SELECT Docstaff,Name,City,State
FROM Docstaff
WHERE ( %ALPHAUP(Name) %STARTSWITH %ALPHAUP( : :Name) )
AND (City %STARTSWITH : :City)
ORDER BY Name
```

This lookup query retrieves rows from the "Docstaff" base table by performing a compound match of the field "Name" against the lookup information variable "::Name" and the field "City" against the lookup information variable "::City". In both cases, the match uses partial matching logic (%STARTSWITH comparison operator). For the field "Name", it performs ALPHAUP case transformation. For all captured rows, the lookup query displays the fields "Name", "City", and "State". It also retrieves the "Docstaff" RowID field. This field is filtered out from the display, but Open M with SQL uses it internally to select a row from the database.

You may customize the default lookup query by editing its WHERE and ORDER BY clauses. For example, to accomplish the following:

1. Restrict the lookup to people living in the state of Massachusetts.

Add a sub-clause to the end of the WHERE clause (using the AND conjunction) that specifies the "State" field must be equal to "Massachusetts".

2. Display the rows in the lookup box sorted first by "City" and then by "Name".

Add the field "City" to the ORDER BY clause, and place it before the field "Name".

The customized lookup query appears as follows:

```
SELECT Docstaff,Name,City,State
FROM Docstaff
WHERE ( %ALPHAUP(Name) %STARTSWITH %ALPHAUP( : :Name) )
      AND (City %STARTSWITH : :City)
      AND (State="Massachusetts" )
```

Index Maps

Every base table has a master map. You can create additional maps for a base table, called Index Maps. This chapter explains the various uses of Index Maps and how to define them. It includes the following topics:

n	Index Maps Speed Row Selection	page 9-2
n	Structure of Index Maps	page 9-3
n	Which Fields Should You Index?	page 9-4
n	Default Structure Index Maps	page 9-5
n	Customized Structure Index Maps	page 9-7
n	Populating an Index Map	page 9-8

Index Maps Speed Row Selection

In a database management system, users generally search for data based on the values of only a few of the fields in the table. The global structure defined in a table's Master Map includes all of the fields in the table; it is usually large and contains much data. Therefore, using a Master Map to search for a particular value of a particular field can involve much reading from disk.

In the Data Dictionary, you can create Index Maps for fields frequently used to retrieve data. Index Maps contain much less information than the Master Map, just the value of the indexed field (or fields) and the corresponding Row ID for that row.

Advantages of Indexing

You can use Index Maps to access rows quickly when you have the value of one or more indexed fields, but do not have the Row ID value for the row containing those fields. In a very large database, the increase in efficiency that results from careful indexing can be significant. In small databases, however, the impact of indexing on speed and efficiency is minimal.

Disadvantages of Indexing

Although indexing can speed access to data, it also results in two penalties:

- It requires additional disk storage, since Open M with SQL creates additional global nodes. See Chapter 16, *Default Physical Structure*, for information on this structure.
- It takes additional time to update when data is added or edited.

For these reasons, you should index only those fields that serve as useful aids in data retrieval and row identification in your application.

Structure of Index Maps

Index Maps define the location of some fields in a base table to another part of the global used by the Master Map, or to a different global.

Index Field and Row ID Values are Subscripts

Generally, Index Maps contain the values of the indexed fields and the Row ID as subscripts; the value of each node is usually NULL. This is always true for Index Maps that Open M with SQL creates under default physical structure. See “Examining a Default Index Map” on page 16-22 in Chapter 16, *Default Physical Structure*, to learn about the global structure of a default Index Map.

Index Maps are Small

Since Index Maps are generally much smaller than Master Maps, there is usually much less I/O involved in locating the row that contains the value of the index field you are trying to match. Global nodes are stored in collating sequence by subscript. It takes fewer disk reads to locate the block which contains the desired node. Once Open M with SQL finds the matching row in the Index Map, it uses the Row ID value there to locate the corresponding row in the Master Map. You now have access to all data in that row.

Rows with Null Value In Index Field Not Included

If a particular row has no data for even one field in an Index Map, Open M with SQL does not insert a corresponding row into that Index Map. This has implications for your application that you should consider.

Example Suppose we want to generate a report containing information about all InterSystems technical writers in order by City in order to come up with car pool possibilities. We use the following query:

```
SELECT *  
FROM Docstaff  
ORDER BY City
```

We defined an index on City but did not make City a required field. If any row does not contain a value for City, that writer will not be included in the report, since Open M with SQL will use the City Index Map to locate rows in the Master Map, and a null index entry cannot exist.

Make Indexed Fields Required to Avoid Null Values

To avoid having an Index Map which does not contain a reference to all the rows in the Master Map, you need to ensure that indexed fields will always contain a value. The best way to do this is to make indexed fields required fields. See

“Defining a Required Field” on page 6-49 in Chapter 6, *Defining Base Table Fields*.

Which Fields Should You Index?

Since Index Maps are useful primarily for fields that Open M with SQL searches through to access a particular row of data, it is useful to have an Index Maps for the following types of fields:

- n Unique fields
- n Fields used to retrieve rows:
 - Lookup fields used for row selection in Form Generator forms
 - Fields used to create frequently used M/PACT reports
 - Fields used in frequently performed queries
- n Fields you sort by

Unique Fields

When you enter a new row which includes a value for a unique field, Open M with SQL looks through all values for that unique field to make sure the new value does not already exist. Since Open M with SQL does this search every time you enter or edit a row that contains a unique field, it is a good idea to have a separate Index Map for each unique field, with the exception of the Row ID field. Unique fields are further described in “Step 5: Specify If Values Must be Unique” on page 6-34 in Chapter 6, *Defining Base Table Fields*.

Fields Used to Retrieve Rows

It is useful to create an Index Map for all fields defined as form lookups, since Open M with SQL searches through the values of these fields, comparing them to the value a user enters on a form, to perform row selection. For further details on lookup fields, refer to the *Open M/SQL Form Generator Guide*.

Similarly, it is useful to define Index Maps for fields used to retrieve rows in M/PACT reports. See the *Open M/SQL M/PACT Guide*.

Example Suppose in Form Generator you create a form for the Docstaff table that includes all of its fields. We define Name to be a lookup field. Therefore, only Doc_Name appears on the Docstaff form row selection window.

To update data about a particular staff person, the user enters the staff person's name at the Docstaff form row selection window. Open M with SQL uses the lookup query it created for that field. The lookup query uses the Data Dictionary's index map for the Name field. The index map contains the Name field and its corresponding Row ID. Open M with SQL then uses this Row ID to locate the

row in the Master Map, and then displays the information on the Docstaff form's master window.

Fields You Sort By

When you sort by a particular field to create an M/PACT report, or within a relational application, Open M with SQL retrieves the data in that field repeatedly from all the rows in the base table. Therefore, fields you sort by are candidates for Index Maps.

Default Structure Index Maps

When you use default physical structure for a base table, Open M with SQL automatically creates Index Maps for these fields:

- n Unique fields
- n Lookup fields

In addition, you can specify that Open M with SQL create Index Maps for additional fields. Specifying default physical structure is described in “Generating Default Physical Structure” on page 5-15 in Chapter 5, *Defining a Base Table* .

Defining Additional Fields to Index

You can create Index Maps for fields other than those which are unique or lookup fields.

Procedure To define additional fields to index:

1. Go to the Base Table Definition master window for the base table for which you want to define additional indexes.
2. Press <RETURN> at the <Physical Structure> branching field
3. This displays the Default Physical Structure window. Press <TAB> to move the cursor to the <Fields to Index> branching field, then press <RETURN>.

```
+-----Base Table Definition-----+
|                                     |
| Base Table Name      Description   |
| Docstaff_____     Information on documentation staff_ |
|                                     |
+-----Default Physical Structure-----+
|                                     |
| Index Fields         |
| City_____          |
| _____           |
| _____           |
| _____           |
| _____           |
| _____           |
| _____           |
| _____           |
| _____           |
|                                     |
+-----+
...Fields to Index On                Press <PF1><PF3> For Help
```

You can index any field in the base table except multi-line fields and the Row ID. You can, however, index constituent fields of a multi-field Row ID. The Row ID is automatically included in every Index Map; you do not need to include it in the index map definition.

4. Press <RETURN> when you are done.
5. Press <RETURN> to move the cursor to the next group of lines to create another Index Map, or press <PROCEED> to complete index map definition.
6. You again see the Default Physical Structure window. This time, press <TAB> to move the cursor to the <Examine Structure> branching field, then press <RETURN>. This generates the Index Map(s) you have defined, assigning each index map a unique name, consisting of the word “Index”, the name of the first field specified for the index map, and an assigned number.

Customized Structure Index Maps

Index Maps perform the same function whether you are using default or customized physical structure. However, if you are using customized physical structure, Open M with SQL will not generate Index Maps automatically.

If you are using customized physical structure to create a new M database, InterSystems recommends that you first use default physical structure when you create your base table fields and lookups, so Open M with SQL generates Index Maps for your unique and lookup fields. You can later switch to customized physical structure and edit the Index Map definitions that Open M with SQL created to match the database structure you have designed.

If you are using customized physical structure to map to an existing M database, you need to define your index maps to match the structure of existing indexes.

See Chapter 17, *Relational Definition of an M Database* and Chapter 18, *Creating a Customized Map Definition*, to learn how to create Index Maps using customized physical structure.

Populating an Index Map

Normally, you would define all index maps when designing a base table, before that table contains any data. However, it is possible to create index maps at any time, and to populate these maps with the existing data in the base table.

The Data Dictionary main menu allows you to fully populate new or modified index maps for a base table that already has defined data. From this menu select the **Populate Index Maps for a Table** option, as shown below:

```

----- OPEN M Developer -----
+-----Data Dictionary-----+
|
| Base Table Definition
| View Definition
| Compile a Table
| Copy a Base Table Field
| Copy a View
| Change View's Starting Table
| Recreate all Default Structure Maps
| Populate Index Maps for a Table
| Reports on Data Dictionary
|
+-----+

Thursday Feb 27, 1997                      Directory: /us/susang/
                                           Copyright (c) 1993 - InterSystems Corporation

Data Dictionary  11:04AM                      Press <PF1><PF3> For Help
Press <RETURN> to (Re)Populate all index maps for a Base Table

```

When you select the **Populate Index Maps for a Table** option, the Data Dictionary displays a Base Table Selection window. Specify the name of the desired base table, and press <RETURN>. This populates all indexes defined for this base table and recompiles the base table, displaying status and run-time error messages as appropriate. You see **DONE** appear in the message display window as each step in the process completes. The messages in the following example are typical of the successful populating of an index map:

```

Populating all indexes for Base Table: Patient
Compiling Base Table...                      DONE
Creating routine to populate indexes...      DONE
Compiling Routine ms2j...                   DONE
Running routine ms2j...
All tuples' indexes updated successfully.
Deleting routine ms2j...                     DONE
Press <RETURN>...

```

Data Conversion and Validation

Data conversion is the process used by Open M with SQL to convert field values from external input format to internal storage format and from internal storage format to external display format.

Data validation is the process used by Open M with SQL to test data values to make sure they meet certain conditions before accepting them.

This chapter describes both data conversion and validation. Specifically, it covers the following topics:

n Overview of Data Conversion and Validation	page 10-2
n Field Conversion Code	page 10-4
n Field Validation Code	page 10-7
n Variables for Field Conversion and Validation Code	page 10-9
n Order of Field Conversion and Validation Events	page 10-10
n Base Table Validation Code	page 10-15

Overview of Data Conversion and Validation

You can manually insert customized conversion and validation code for any field at the field level. You can also insert validation code at the base table level.

If defined at the base table level, the conversion and/or validation code acts as a default for all forms associated with that table.

If defined at the form level, the conversion and/or validation code overrides the default base table code for that particular form only.

Data Conversion

Conversion code applies to the field definition level only. It converts the data value of a field from an external input value to an internal storage format and from an internal storage format to an external display format.

Data Validation

The Data Dictionary lets you insert M code to perform data validation at two levels:

- n Row level
- n Field level

At the row level, validation code sets conditions that must be met to validate a database operation (insert, update, or delete) on an entire row. This is called base table validation code.

At the field level, validation code sets conditions that must be met to validate data values for a particular field.

Fields Can Have Internal and External Values

A field's internal value is the value stored in the database. Open M with SQL treats the internal value as the actual value of the field. The internal value is used in SQL conditions and calculations. It is also the value tested by field validation code.

A field's external value has two formats:

- n External Input Value - value entered by the user in a form or SQL statement.
- n External Display Value - value displayed by Open M with SQL in the output of a form, report, or query.

It often makes sense to use internal storage values that differ from external values. Internal values can strip away formatting punctuation and store data in a

simple abbreviated fashion. This conserves database storage space. It also allows you to input and/or display data values in different formats. For example, Open M with SQL stores dates internally as \$HOROLOG strings but represents them externally in a variety of formatting patterns.

Automatically Generated Conversion and Validation Code

For most data types, Open M with SQL automatically generates conversion and validation code to perform basic conversion and validation functions.

The following table shows the types of code (External-to-Internal, Internal-to-External, and Validation) Open M with SQL automatically generates for each data type:

Table 10-1: Automatically Generated Code by Data Type

Data Type	Ext-Int Conversion	Int-Ext Conversion	Validation
Date	4	4	4
Designative Reference	4		
Multiple Choice	4	optional	
Name			4
Number	4	optional	4
Row ID	4		4
Text			
Time	4	4	4
Yes/No	4	4	4

For the data types Date, Time, and Number, the conversion and validation code that Open M with SQL generates depends on what range, format, and display selections you make in the Data Type window. For more information, see Chapter 6, *Defining Base Table Fields*.

You can add to, modify, or delete automatically generated conversion code (External-to-Internal and Internal-to-External). You can add to automatically generated validation code, but cannot modify or delete the automatically-supplied code.

Field Conversion Code

For some data types, such as date and time fields, Open M with SQL automatically generates conversion code. This code is completely modifiable. You may enhance and customize it, or you may delete it.

For data types where Open M with SQL does not automatically generate any conversion code, you may define such code manually.

There are two types of conversion code: external-to-internal, and internal-to-external. The external value of a field is the value entered by the user in a form or SQL statement. External-to-internal conversion code acts on the external input value to convert it to an internal storage value. Internal-to-external conversion code then acts on the internal storage value to convert it to an external display value that appears in the output of forms, reports, and queries.

External-to-Internal Conversion Code

Open M with SQL lets you insert M code to convert the external input value (user input) of a field into an internal storage format. This External-to-Internal Conversion Code is often used to conserve database storage space and to make input easier for users.

For each data type, Open M with SQL automatically generates External-to-Internal conversion code that does the following:

Table 10-2: Automatically Generated External-to-Internal Conversion Code

Data Type	External-to-Internal Conversion Code
Date	Makes sure the user enters the external input value in one of the seven possible date formats. If not, it displays an error message. Converts the value to \$HOROLOGY date format for internal storage.
Designative Reference	This is a non-displayable field. Users cannot enter data directly into it. It mimics the Row ID field of the designated table. The code makes sure the value is a pure number (i.e., contains no non-numeric characters). If not, it displays an error message.
Multiple Choice	Converts the user-selected value to the specified internal value. If internal values are not defined, it simply stores the selected value.
Name	None
Number	Makes sure the external input value entered by the user is either: 1. A pure number (i.e., contains no non-numeric characters), or 2. A number entered in the number format selected in the Number Data Type window. If not, it displays an error message. Strips away all punctuation.

Table 10-2: Automatically Generated External-to-Internal Conversion Code

Data Type	External-to-Internal Conversion Code
Row ID	This is a non-displayable field, and users cannot enter data into it (unless you use customized mapping). Its value is derived from an internal counter. The code makes sure the value is a pure number (i.e., contains no non-numeric characters). If not, it displays an error message.
Text	None
Time	Makes sure the users enters the external input value in one of the four possible time formats. If not, it displays an error message Converts the value to \$HOROLOG time format for internal storage.
Yes/No	Converts the user-selected value to Y or N for internal storage.

Open M with SQL places the external input value into the special variable, %val. External-to-Internal Conversion Code consists of M statements that convert the value of %val into the internal storage value.

Optionally, it can also perform validation tests on the external input value, as described in “Field Validation Code” on page 10-7.

You cannot reference base table field names in External-to-Internal Conversion Code.

Example The following External-to-Internal Conversion Code converts a date from external to internal (\$H) format:

```
SET %DS=%val DO INT^%DATE SET %val=%DN
```

The entry point INT^%DATE takes its input from the variable %DS and takes its output from the variable %DN.

Internal-to-External Conversion Code

Open M with SQL lets you insert M code to convert internal storage values into modified external display values. This is called Internal-to-External Conversion Code.

Chapter 10—Data Conversion and Validation

For each data type, Open M with SQL automatically generates Internal-to-External conversion code that does the following:

Table 10-3: Automatically Generated Internal-to-External Conversion Code

Data Type	Internal-to-External Conversion Code
Date	Converts the internal \$HOROLOG date value to the display format selected in the Date Data Type auxiliary window.
Designative Reference	None
Multiple Choice	Converts the internal storage value to the external display value. If internal values are not defined, this code is not present.
Name	None
Number	Converts the internal storage value to the external display format selected in the Number Data Type auxiliary window, including decimal places, leading character punctuation, and negative number display style.
Row ID	None
Text	None
Time	Converts the internal \$HOROLOG time value to the display format selected in the Time Data Type auxiliary window.
Yes/No	Converts the internal value to an external display value of "Yes" or "No".

Internal-to-External Conversion Code consists of M statements that modify the value of the Open M with SQL variable %val. Internal-to-External Conversion Code finds the internal value in %val and then sets %val to the external display value.

In addition to not being able to reference field names in Internal-to-External Conversion Code, the variables %ok and %msg have no effect.

Example The following Internal-to-External Conversion Code converts a date from internal to external format:

```
SET %val=$ZDATE(%val)
```


Field Validation Code

Field validation code establishes validation standards for the value of a field so as to prohibit inappropriate entries. Open M with SQL invokes validation code on the converted internal value of a field after the external-to-internal conversion is complete and before the internal-to-external conversion is invoked.

Open M with SQL automatically generates field validation code for most data types. You cannot modify or delete automatically generated validation code. It is a permanent attribute of the data type you have selected. However, you can define Additional Validation Code to enhance the automatically generated validation code.

For each data type, Open M with SQL automatically generates Validation code that does the following:

Table 10-4: Automatically Generated Validation Code

Data Type	Validation Code
Date	Makes sure the converted internal value is a number. If not, it displays an error message. Makes sure the converted internal value falls within the range specified in the Date Data Type auxiliary window. If not, it displays an error message.
Designative Reference	None
Multiple Choice	None
Name	Makes sure the user entered value starts with a letter and has the format: O'Last,First Middle Jr. If not, it displays an error message.
Number	Makes sure the converted internal value is a pure number (i.e., contains no alphabetic characters). If not, it displays an error message. Makes sure the converted internal value falls within the range specified in the Number Data Type auxiliary window. If not, it displays an error message. Makes sure the converted internal number does not have more than the specified number of decimal places. If not, it displays an error message.
Row ID	None
Text	None

Table 10-4: Automatically Generated Validation Code

Data Type	Validation Code
Time	Makes sure the converted internal value is a number. If not, it displays an error message. Makes sure the converted internal value falls within the range specified in the Time Data Type auxiliary window. If not, it displays an error message.
Yes/No	Makes sure the converted internal value is equal to "Y" or "N". If not, it displays an error message.

Additional Validation Code

Additional Validation Code allows you to refine the scope of allowable entries for a particular field beyond that defined in the automatically generated validation code. Additional Validation Code consists of M statements that test the value of %val and accept or reject an entry based on the results of those tests.

At validation time, %val holds the internal value of a field. You can use Additional Validation Code to reject an invalid value for %val by setting the variable %ok to zero. You can also display an error message by setting the variable %msg to a text error message.

Example For example, if you have a Date field and you want to prevent users from entering any future dates, you could insert the following Additional Validation Code:

```
IF %val>$H SET %ok=0,%msg="No future dates please."
```

This code specifies that any date later than the current date (the value of \$H) fails the validation test. When this occurs, the message "No future dates please" will display on the screen.

Validation Checking on External Values

Validation code acts on the internal value of a field. If you want to perform validation checking on the original external value, i.e., the format in which the user entered the value, you must include pre-conversion validation code inside the External-to-Internal Conversion Code.

You may write validation tests into the External-to-Internal Conversion Code using the same syntax as described for Additional Validation Code. The only difference is the order in which it is processed.

You cannot write validation tests into the Internal-to-External Conversion Code.

Variables for Field Conversion and Validation Code

Open M with SQL supports several variables used for field-level conversion and validation code:

Table 10-5: Open M with SQL Conversion and Validation Variables

Variable	Function
%val	Stores the current value of a field.
%ok	Flag that can be set to 0 to disallow the filing of an invalid field value.
%msg	Error message text for screen display when rejecting an invalid field value.
%data(<i>col</i>)	Stores the data value of the field of the current row with the column number equal to the value of <i>col</i> , whether saved or unsaved.
%edit(<i>col</i>)	Stores the disk value of the field of the current row with the column number equal to the value of <i>col</i> , when that field has been edited but not saved. Once %data is saved, %edit is null.
%linenum	Used only for multi-line fields (Text data type). Specifies the line number within the field of the line on which the cursor is currently located. Multi-line fields are described in "Step 7: Specify If the Field is Multi-Line" on page 6-37 in Chapter 6, <i>Defining Base Table Fields</i>

You cannot reference field names in field conversion or validation code.

You also cannot reference the following Open M with SQL variables in field conversion or validation code:

- n %old
- n %oldext
- n %new
- n %newext

Order of Field Conversion and Validation Events

When a user enters data into a field, Open M with SQL may convert the data value to an internal storage value and may perform validation checks on the data value. The following is the sequence of events for field conversion and validation:

1. User enters a value into a field.
2. Open M with SQL invokes External-to-Internal Conversion Code.

This code may first perform validation checks on the external value and issue an error message if the data value is invalid. This validation code may be system-supplied or user-written. This code then converts the external data value to the internal storage value. This conversion code may be system-supplied or user-written.

3. Open M with SQL invokes Validation Code.

This validation code is supplied by the Data Dictionary and cannot be user modified. It performs validation checks on the internal value. These validation checks are specific to the data type. The user may see an error message to indicate that s/he has attempted an invalid entry.

4. Open M with SQL invokes Additional Validation code.

This validation code is supplied by the user to perform further validation checks on the internal storage value. The user may see an error message to indicate that s/he has attempted an invalid entry.

5. Open M with SQL stores the valid value in internal storage format.
6. The user retrieves the stored value.
7. Open M with SQL invokes External-to-Internal Conversion Code and then displays the converted value in the proper external format. This conversion code may be system-supplied or user-written.

Modifying Field Conversion and Validation Code

The following is an example of Automatically Generated Conversion/Validation Code. When you select Date as the Data Type for a field, you see the Date Data Type popup window, as shown below:

-----Base Table Definition-----	
Base Table Name	Description
Docstaff_____	Information on documentation staff_
-----Field Definition-----	
Field Name	Description
Date_of_hire_____	Date Writer was Hired_____
-----Date - Data Type-----	
Date Format: 6_	Display: October 18, 1966
First Date: January 01, 1990__	Display Full Year: No__
Last Date: _____	Override Display Delimiter: _
Formats: 1 - 10/18/90 5 - Oct 18, 1990 2 - 18 Oct 90 6 - October 18, 1990 3 - 18.10.90 7 - 90.10.18 4 - 18/10/90	
Up+	onal
...Date Data Type Unsaved Data Press <PF1><PF3> For Help	

The Date Data Type window requests that you select one of seven formats for external representation of dates. Format #1 is the default format. In this example, we have selected Format #6 (October 18, 1990).

The Date Data Type window also lets you enter the top and bottom limits for a range of acceptable dates. This is an optional feature. In this example, we have entered a bottom limit of 1/1/90.

Open M with SQL automatically generates conversion and validation code to accommodate the date format and range that you specified for the field.

You can view and edit the automatically generated conversion and validation code by selecting the Conversion/Validation Code option from the horizontal

options menu. This invokes the Conversion/Validation Code window, as shown below:

-----Base Table Definition-----

Base Table Name	Description
Docstaff	Information on documentation staff_

-----Conversion/Validation Code-----

The Field's Value is stored in the Variable '%val'.
The M code below should convert '%val' and store the result in '%val'
e.g. SET %val=\$\$ABC^ROUTINE(%val)

Internal to External M Code (1/2)
s %val=+\$g(%val) s:%val<1 %val="" q:%val=""

##vendor(daten6)

External to Internal M Code (0/1)
s %val=\$\$dateint^%razdt(%val,6,"") s:%val<0 %ok=0

M Code to Validate Internal Value (0/2)
i %val'=\$%val s %ok=0,%msg=\$\$setmsg^%razset("DTDatIn")

s:%val<54422 %ok=0

Additional M Validation Code (0/0) (If Invalid - SET %ok=0)

...Conversion Code Unsaved Data Press <PF1><PF3> For Help

- n You can alter Automatically Generated Conversion Code

Where automatically generated conversion code already exists, you may enhance or delete that code. If no code exists, you may create it. This conversion code is shown on the Internal to External M Code and External to Internal M Code lines in Conversion/Validation Code window.

- n You cannot alter Automatically Generated Validation Code

Where automatically generated validation code exists, you cannot modify or delete it. It is a permanent attribute of the data type you have selected. This validation code is shown on the M Code to Validate Internal Values lines in Conversion/Validation Code window.

- n You can define Additional Validation Code to enhance automatic validation checking.

This validation code is shown on the Additional M Validation Code lines in Conversion/Validation Code window.

Procedure To view and/or customize field conversion and validation code:

1. Enter the Field Definition window for the field whose conversion/validation code you want to view and/or enhance.
2. From the Field Definition window, press the <Go to Bottom Menu> key to access the Field Definition horizontal options menu.

Modifying Field Conversion and Validation Code

- From the Field Definition horizontal options menu, select the Conversion/Validation Code option.

You see the Conversion/Validation Code window, as shown below:

-----Base Table Definition-----	
Base Table Name	Description
Docstaff	Information on documentation staff
-----Conversion/Validation Code-----	
The Field's Value is stored in the Variable '%val'. The M code below should convert '%val' and store the result in '%val' e.g. SET %val=\$\$ABC^ROUTINE(%val)	
Internal to External M Code (1/0)	
External to Internal M Code (0/)	
M Code to Validate Internal Value (0/)	
Additional M Validation Code (0/0) (If Invalid - SET %ok=0)	
...Conversion Code Unsaved Data Press <PF1><PF3> For Help	

- Move the cursor to the Internal-to-External Code field, and insert the appropriate code.

Note: Internal-to-External Conversion Code consists of valid M statements that convert %val from an internal storage value to an external display format.

- Move the cursor to the External-to-Internal Code field, and insert the appropriate code.

Note: External-to-Internal Conversion Code consists of valid M statements that convert %val to an internal storage format.

- Move the cursor to the Additional Validation Code field, and insert the appropriate code.

Note: Additional Validation Code consists of valid M statements that test the internal format of %val to make sure it meets certain conditions. To reject an invalid entry, set %ok=0 and set %msg to a text error message.

- When you have finished inserting your field conversion/validation code, press <PROCEED> to save your definitions and exit the Conversion/Validation Code window.

Chapter 10—Data Conversion and Validation

Example Suppose you want to create a telephone number field.

You are willing to accept user input in any format so long as it contains ten numbers and no alphabetic characters. This means formats such as the following are all appropriate:

```
( 617 ) 621-0600
617-621-0600
6176210600
```

If it does not meet these criteria, you want to disallow the entry and send an error message to the user.

For internal storage, you want to strip all punctuation and convert the data value to pure numeric format.

For external display, you want to convert the value to the following format:

```
( 617 ) 621-0600
```

To do this, you need to define a field of Data Type Text with a maximum length of 14. And, you need to manually insert the following conversion and validation code:

External-to-Internal Conversion Code

The following code strips the original external value (user input) of all punctuation:

```
SET %val=$TRANSLATE(%val,"()-","")
```

Validation Code

The following code checks the punctuation-stripped internal value to make sure that it contains exactly ten numbers and nothing else. If it does not, the code disallows the entry and prints the error message "Invalid Phone Number" to the screen.

```
IF %val'?10N SET %ok=0, %msg="Invalid Phone Number"
```

Internal-to-External Conversion Code

The following code converts the punctuation-stripped internal value to an external format with parentheses around the area code, a space before the telephone prefix, and a hyphen between the prefix and the rest of the number:

```
SET %val="( ${E(%val,1,3)} ) _${E(%val,4,6)} -
_${E(%val,7,10)}
```


Base Table Validation Code

At the base table definition level, you can insert M code to specify conditions that must be met before an insert, update, or delete operation can be performed on a row of data. Usually, this condition defines a constraint among field values in a row.

For example, if you keep a table for department budget planning in which each field represents a different budgeted item, you can use Base Table Validation Code to specify that a row not be filed if the sum of the values of those fields exceeds the value of the field for Total Amount Budgeted. Optionally, you can specify a message that should be displayed if such a violation occurs.

Procedure To define base table validation code:

1. Enter the Base Table Definition window.
2. From the Base Table Definition window, press the <Go to Bottom Menu> key to access the Base Table Definition horizontal options menu.
3. From the Base Table Definition horizontal options menu, select the Validation Code option.

You will see the Base Table Validation window:

-----Base Table Definition-----	
Base Table Name	Description
Docstaff_____	Information on documentation staff_
Use Default Physical Structure? Yes_	
+-----Base Table Validation-----+	
Enter lines of M code to execute before performing each operation.	
To abort the operation - SET %ok=0	
To display an error message - SET %msg="Your Message"	
Insert Validation M Lines (1/0)	

Update Validation M Lines (0/)	

Delete Validation M Lines (0/)	

+-----+	
...Base Table Validation	
Press <PF1><PF3> For Help	

Chapter 10—Data Conversion and Validation

The following table lists and describes the fields located on the Base Table Validation window:

Table 10-6: Fields on Base Table Validation window

Field Name	Description
Insert Validation M Lines	Here you may enter M code to specify conditions that must be met before a new row can be inserted into the database.
Update Validation M Lines	Here you may enter M code to specify conditions that must be met before an existing database row can be updated.
Delete Validation M Lines	Here you may enter M code to specify conditions that must be met before a database row can be deleted.

4. Enter M code in the appropriate field(s) to place conditions on database operations.
5. When you have finished inserting your base table validation code, press <PROCEED> to save your definitions and exit the Base Table Validation Code window.

Defining Base Table Validation Code

Unlike field-level conversion and validation code, Base Table Validation Code lets you reference field names enclosed in curly braces such as {fieldname}. You cannot refer to multi-line fields using the curly brace syntax.

To disallow filing of a row if the specified condition is not met, set the variable %ok equal to zero, as in:

```
IF {a}+{b}'={ab_total} SET %ok=0
```

To send an error message to the user, set the variable %msg to the text of the error message, as in:

```
IF %ok=0 SET %msg="Invalid Data"
```

The error message displays whenever a violation occurs.

Example This example shows how to define base table validation code to prevent insert of a row into the database when the sum of the fields "Salaries", "Supplies", and "Miscellaneous" exceeds the value of the field "Total Budgeted Amount".

When a violation of this condition occurs, the message "Total of budgeted items exceeds budgeted amount" will display.

Field names referenced in the validation code are enclosed in curly braces.

-----Base Table Definition-----	
Base Table Name	Description
Budget	Department Budgets
Use Default Physical Structure? Yes_	
-----Base Table Validation-----	
Enter lines of M code to execute before performing each operation.	
To abort the operation - SET %ok=0	
To display an error message - SET %msg="Your Message"	
Insert Validation M Lines (2/1)	
i {Salaries}+{Supplies}+{Miscellaneous}>{TotalBudget} s %ok=0	
i %ok=0 s %msg="Total of budgeted items exceeds budgeted amount."	
Update Validation M Lines (0/0)	
Delete Validation M Lines (0/0)	

<div> <div>...Base Table Validation</div> <div>Press <PF1><PF3> For Help</div> </div>	

Base Table Triggers

Open M with SQL provides a simple and powerful mechanism for programming the automatic execution of database operations in an application. This mechanism is the ability to define **triggers**. Triggers are sequences of actions which you may define to automatically take place at various points during the execution of your Open M with SQL application.

You can associate triggers with base tables or with forms. This chapter begins with a general overview of all Open M with SQL triggers, then describes in detail how to define and use base table triggers. Specifically, it covers the following topics:

- n Overview of Open M with SQL Triggers page 11-2
- n Overview of Base Table Triggers page 11-4
- n Base Table Trigger Code page 11-7
- n Examples of Base Table Triggers page 11-15

Overview of Open M with SQL Triggers

Generally speaking, there are two kinds of triggers in Open M with SQL, **base table triggers** and **form triggers**.

Base table triggers are sequences of database actions initiated by an INSERT, UPDATE, or DELETE action performed on a base table. These triggers help enforce integrity constraints and other data dependencies. Base table triggers perform operations tailored to the needs of the database manager. For example, a Pre-Filing DELETE trigger may use SQL statements to insert the information into an archive file, or a Post-Filing INSERT trigger in a patient database may conditionally send a report to the head of the cardiology department if a patient is admitted with severe chest pains.

Form triggers extend the conventional notion of triggers to forms, windows, and fields. They execute at different stages of your application and help define its control structure by linking together the various capabilities of an application and establishing a logical flow of events. Form triggers perform operations tailored to the needs of the application end-user.

You Can Associate Triggers with Different Kinds of Objects

Open M with SQL lets you define triggers associated with any of the following objects:

- n Base Table
- n Form
- n Window
- n Field (on a form)

Triggers invoked from forms, windows, or fields are collectively referred to as form triggers because they are tied to forms or form objects.

Triggers invoked from base tables are referred to as base table triggers.

The following table shows what types of triggers you can define for each trigger-capable Open M with SQL object:

Table 11-1: Types of Triggers

Object	Trigger Types
Base Table	Pre-Filing INSERT Pre-Filing UPDATE Pre-Filing DELETE Post-Filing INSERT Post-Filing UPDATE Post-Filing DELETE
Form	Pre-Form Post-Retrieval Post-Form
Window	Pre-Window Post-Window
Field	Pre-Field Post-Field

The remainder of this chapter addresses base table triggers. For more information on form triggers, see the *Open M/SQL Form Generator Guide*.

Overview of Base Table Triggers

Base table triggers are useful primarily for operations that preserve the integrity of the database. They specify that some database modification or other processing should occur in response to an INSERT, UPDATE, or DELETE operation performed on the database.

You should be careful not to confuse base table triggers with base table validation code. It is not the job of base table triggers to determine whether an INSERT, UPDATE, or DELETE operation should or should not be performed -- that is the job of base table validation code. Base table triggers define:

- Additional actions to be performed in concert with the INSERT, UPDATE, or DELETE operation.
- If you use customized mapping, how to file fields for which Open M with SQL does not generate filing code due to how you mapped their location. See Chapter 18, *Creating a Customized Map Definition*, to learn the circumstances under which Open M with SQL does not generate filing code for a field.

You can program base table triggers to execute either before filing occurs or after filing occurs.

Base table triggers can execute SQL code, M code, or an M routine.

Pre-Filing Triggers

Pre-Filing triggers are invoked after the user presses the save key to indicate that a row will be saved but before the row is actually filed to the database.

The following table lists the three types of Pre-Filing triggers and describes when each is invoked and what the status of the database is at the time of invocation:

Table 11-2: Pre-Filing Base Table Triggers

Trigger Type	When Invoked	Status of Database
INSERT	When a new row is inserted	Row not yet inserted
UPDATE	When an existing row is edited	Row not yet updated
DELETE	When an existing row is deleted	Row not yet deleted

Pre-Filing triggers are useful when you want to update a field in the current base table in concert with an INSERT, UPDATE, or DELETE operation. In this case,

you must define a Pre-Filing trigger to make sure the update produced by the trigger is filed along with the rest of the table. Once Open M with SQL has filed the current table, it will not file it again to preserve updates produced by a Post-Filing trigger.

Open M with SQL does not permit you to update a field by SETting it in a trigger. If you wish to update a field with a Pre-filing trigger, you can perform this operation as follows:

```
%edit(2)=%data(2), %data(2)=value
```

In this example, the 2 identifies the field (in this case, the second column of the row), and *value* is the new data value for the field.

Post-Filing Triggers

Post-Filing triggers are invoked immediately after a row has been filed to the database.

The following table lists the three types of Post-Filing triggers and describes when each is invoked and what the status of the database is at the time of invocation:

Table 11-3: Post-Filing Base Table Triggers

Trigger Type	When Invoked	Status of Database
INSERT	When a new row is inserted	Row already inserted
UPDATE	When an existing row is edited	Row already updated
DELETE	When an existing row is deleted	Row already deleted

Post-Filing triggers are useful when you want to update fields in other tables in concert with an INSERT, UPDATE, or DELETE operation on the current table. Post-Filing triggers allow you to make sure that the filing actually transacts before initiating the update. Open M with SQL still files updates made to other tables even after the current table has already been filed.

SQL Code Triggers

You can define base table triggers to invoke SQL code. SQL code triggers are useful when you want to perform an operation on the database, such as an INSERT, UPDATE, or DELETE.

Chapter 11—Base Table Triggers

Examples of SQL triggers include the following:

- n If you have a Designative Reference field from the Employees base table into the Departments base table, you might maintain a field in the Departments base table to keep the number of employees in the department. You can define a trigger that automatically increments or decrements the number of employees counter in the Department base table every time an employee leaves or joins the organization, or changes departments.
- n You may have an Employees/Departments schema in which a Designative Reference field Department_Manager in the Departments base table designates a row in the Employee base table. You may want to define a trigger that performs an update in both tables if an employee who is also a department manager leaves the organization.

M Code Triggers

You can define base table triggers to invoke M code. M code triggers are useful when you want to enhance the information flow in your application by manipulating numbers, performing calculations, or sending messages.

The most common use of an M code trigger is to send a message to a user or group of users when a particular database action is performed.

M Routine Triggers

You can also define base table triggers to invoke an M routine. This lets you make changes to the routine rather than changing the directly inserted code, which forces you to recompile the base table.

Base Table Trigger Code

When creating an application, you may decide that a base table trigger is the best way to enforce a particular integrity constraint.

You can define as many base table triggers as you want. In addition, Open M with SQL defines one base table trigger automatically. This automatically-defined trigger deletes child rows when a parent row is deleted. For more information about this trigger, see the section of this chapter entitled, “Automatic Trigger Deletes Child Table Rows” on page 11-14.

You can check that all triggers that you defined are valid by running the Object Integrity Checker Utility, described in the *Open M/SQL Developer Guide*.

Trigger Items

A trigger may consist of one trigger item or a sequence of multiple trigger items. When a trigger is invoked, the list of trigger items executes sequentially from top to bottom in the order defined. Optionally, you can make the execution of each trigger item conditional upon an M expression, in which case the trigger items are evaluated in sequence but execute only when the specified condition(s) are satisfied.

Trigger Action Types

Base table triggers can invoke any of the following actions:

- n SQL code
- n M code
- n Routine

Conditional Execution of Triggers

You can optionally make the execution of any trigger item conditional upon the truth of an M expression.

If you define a condition in the IF (M Expression) field on the Trigger Definition window and that condition evaluates to true, the corresponding trigger item will execute. If the condition evaluates to false, the corresponding trigger item will not execute.

You may define an unconditional trigger item by leaving the IF (M Expression) field blank. In this case, the trigger item always executes.

Open M with SQL supports a series of percent variables which may prove useful in constructing M expression conditions for trigger execution. For more informa-

Chapter 11—Base Table Triggers

tion on these percent variables, see the section of this chapter entitled “%-Variables in Base Table Triggers” on page 11-8.

Execution of Multiple Trigger Items

For a given trigger, the list of trigger items executes sequentially from top to bottom. Conditions established via the IF (M Expression) field are evaluated separately for each trigger item defined. However, you may define the execution of a trigger item to be conditional upon the outcome of a previously defined conditional trigger item.

You May Reference Field Values in Base Table Triggers

To reference a field in base table triggers, enclose the field name in curly braces. For INSERT and UPDATE triggers, the field enclosed in curly braces refers to the new field value. For DELETE triggers, the field enclosed in curly braces refers to the old value.

%-Variables in Base Table Triggers

Open M with SQL uses the %-variables %data and %edit to hold new and existing values for fields that are being edited. You may find it useful to reference these %-variables in base table triggers.

The %data array stores the data values for fields in the current row of the current table. When a field value is edited, the old value is stored in the %edit array and the new value is stored in the %data array. When the row is filed, the value of %data overwrites the current value, and %edit is nullified.

InterSystems discourages the direct use of the %data and %edit arrays in SQL triggers due to the possibility of their being NEWed within the SQL trigger. Use a previous trigger item to copy the values from %data and %edit into local variables.

Column Number Identifies Field

The array subscript for %data and %edit is the column number for the field whose value you want to access. To determine a field's column number, look in the Update Features window of the field's definition.

Example If the field Title has a column number of 5, you can reference its current value as follows:

```
%data(5)
```

or you can reference its old value as follows:

```
%edit(5)
```

Values of %data and %edit Depend on Trigger Type

The %data and %edit arrays hold different values depending on the type of trigger you are defining. The following tables illustrates the pattern for these values:

Table 11-4: Values Held By %data and %edit for Different Trigger Types

Trigger Type	%data	%edit
INSERT	new value	undefined
UPDATE	new value	old value
DELETE	undefined	old value

%data and %edit always have the same value for both Pre-Filing and Post-Filing triggers.

Defining a Base Table Trigger

Procedure To define a base table trigger:

1. Enter the Base Table Definition window.

The Base Table Definition window is shown below:

-----Base Table Definition-----							
Base Table Name		Description					
Budget_____		Department Budgets_____					
Use Default Physical Structure? Yes_							
< Fields >							
< M/WINDOWS Lookups >							
< Physical Structure >							
Approximate Number of Rows 250_____							

<div> <div>Base Table Definition</div> <div>Unsaved Data</div> <div>Press <PF1><PF3> For Help</div> </div>							
Implicit Joins	Validation Code	BaseTable Triggers	Compile Table	Comp Rel Objects	Copy Field	Advanced Options	Comments

2. Press the <Go To Bottom Menu> key to access the Base Table Definition horizontal options menu.
3. On the Base Table Definition horizontal options menu, select the BaseTable Triggers option.

The Base Table Triggers window appears, as shown below:

-----Base Table Definition-----							
Base Table Name		Description					
Budget		Department Budgets					
Use Default Physical Structure? Yes_							
< Fields >							
-----Base Table Triggers-----							
--- Pre Filing ---				--- Post Filing ---			
< INSERT >				< INSERT >			
< UPDATE >				< UPDATE >			
< DELETE >				< DELETE >			
...Base Table Triggers Unsaved Data Press <PF1><PF3> For Help							
Implicit	Validation	BaseTable	Compile	Comp Rel	Copy	Advanced	Comments
Joins	Code	Triggers	Table	Objects	Field	Options	

- In the Base Table Triggers window, use the cursor keys to navigate and press <RETURN> to select the appropriate trigger type option.

The Trigger Definition window appears, as shown below:

-----Base Table Definition-----							
Base Table Name		Description					
Budget		Department Budgets					
Use Default Physical Structure? Yes_							
< Fields >							
-----Trigger Definition-----							
Action	Form/Window/Field	(M Expressions)					
Type	Query/Routine						
_____	_____	IF	_____				
_____	_____	IF	_____				
_____	_____	IF	_____				
_____	_____	IF	_____				
_____	_____	IF	_____				
Implicit Validation BaseTable Compile Comp Rel Copy Advanced Comments							
...Trigger Definition Press <PF1><PF3> For Help							

Chapter 11—Base Table Triggers

The following table lists and defines the fields located on the Trigger Definition window:

Table 11-5: Trigger Definition Window

Field Name	Description
Action Type	This is a required field. You may select any of three possible values <ul style="list-style-type: none">n M Coden SQLn Routine Press the <List Choices> key to see a lookup box that lists these choices.
Form/Window/Field/Query/Routine	This field becomes required when you select Routine as the Action Type. In all other cases, it is skipped. When required, you must enter a valid routine name. Use the following syntax: ^routine or tag^routine
(M Expressions)	Here you can optionally specify an IF condition in the form of an M expression. The associated trigger action executes only if the condition is true. If no condition is specified, the trigger always executes.

5. In the multi-row Trigger Definition window, enter an Action Type for your first trigger item and press <RETURN>.
 - a. If you select M Code action type, the M Code Lines popup window appears, as shown below:

```

+-----Base Table Definition-----+
Base Table Name          Description
Budget_____          Department Budgets_____

      Use Default Physical Structure? Yes_

< Fields >
+-----Trigger Definition-----+
      Action      Form/Window/Field
      Type        Query/Routine          (M Expressions)

M Code_____      IF _____
_____            IF _____
_____            IF _____
_____            IF _____
+-----M Code Lines-----+
M Code Lines (1/0)
_____
_____
+-----+
...M Code Lines          Unsaved Data          Press <PF1><PF3> For Help

```

Here, you may type the M code that you want the trigger to execute. Press <PROCEED> when you are done.

- b.** If you select the action type SQL, the SQL Lines popup window appears, as shown below:

```

+-----Base Table Definition-----+
Base Table Name          Description
Budget_____          Department Budgets_____

Use Default Physical Structure? Yes_

< Fields >

+-----Trigger Definition-----+
Action      Form/Window/Field
Type        Query/Routine          (M Expressions)

SQL_____          IF _____
_____          IF _____
_____          IF _____
_____          IF _____

+-----SQL Lines-----+
+ SQL Lines (1/0) _____
+ _____
+ _____
+-----+

...SQL Lines          Unsaved Data          Press <PF1><PF3> For Help

```

Here, you may type the SQL code that you want the trigger to execute. Press <PROCEED> when you are done.

- c. If you select action type Routine, you can enter the name of a routine that you want the trigger to invoke. Enter the routine name in the Form/Window/Field/Query/Routine field and press <RETURN>.

Note: In the IF (M Expression) field, you may optionally enter M code that sets a condition which must prove true in order for the trigger item to execute. Press <RETURN> when you are done.

6. You may enter as many trigger items as you like. When you are done, press <PROCEED> to save your definitions and exit back to the Base Table Trigger window.
7. You may define more triggers by selecting another choice, or you may press <PROCEED> to exit back to the Base Table Definition window.

Automatic Trigger Deletes Child Table Rows

If you have a characteristic relationship between base tables such as an Invoice/Line Items relationship, Open M with SQL automatically generates a Pre-Filing DELETE trigger in the parent table that deletes dependent line items in the child table every time the parent invoice is deleted. If you delete the parent table, this trigger automatically deletes the child table.

This is an ordinary trigger in every sense. It is visible as a trigger item in the Pre-Filing DELETE Trigger Definition window of the parent table. You can modify it, or even delete it.

Open M with SQL defines the automatic delete trigger to be Pre-Filing as a precaution to make sure that it can delete the children before it deletes the parent.

If you edit the name of the parent reference field in a child table, Open M with SQL automatically updates the Pre-Filing DELETE trigger in the parent table to match the edited field name.

Example Shown below is the automatic delete trigger generated for a characteristic relationship between the Guides parent table and the Editions child table:

-----Base Table Definition-----			
Base Table Name		Description	
Guides_____		Titles of Intersystem Guides_____	
Use Default Physical Structure? Yes_			
< Fields >			
-----Trigger Definition-----			
Action Type	Form/Window/Field Query/Routine	(M Expressions)	
SQL_____	_____	IF _____	_____
_____	_____	IF _____	_____
_____	_____	IF _____	_____
_____	_____	IF _____	_____
-----SQL Lines-----			
+ SQL Lines (1/1)			
DELETE FROM Editions WHERE Guides=:%data(1)_____			
+_____			
+_____			
+_____			
...SQL Lines		Press <PF1><PF3> For Help	

Examples of Base Table Triggers

Example of Routine Action Type Trigger

This is an example of a routine action type. It shows how we automatically update the value of Latest_Edition_Gui in the Guides table. Remember, this is an output only field, which is a designative reference to the Editions table. We created Post-Filing INSERT and UPDATE triggers in the Editions table which check to see if an edition being inserted is the latest edition. If it is, it places the Row ID value of the edition in the Latest_Edition_Gui for the row in Guides which is the Parent Reference for the row being inserted in Editions.

The trigger calls an M routine, LatestEd, with two arguments: the values of the fields Editions (the Row ID, which is {Parent Reference||childsub}) and Revision_Date:

-----Base Table Definition-----		
Base Table Name	Description	
Editions_____	Editions of Guides_____	
Use Default Physical Structure? Yes_		
< Fields >		
-----Trigger Definition-----		
Action Type	Form/Window/Field Query/Routine	(M Expressions)
M Code_____	IF _____	_____
_____	IF _____	_____
_____	IF _____	_____
_____	IF _____	_____
-----M Code Lines-----		
M Code Lines (1/1)		
d ^LatestEd({Editions},{Revision_Date})_____		

...M Code Lines Unsaved Data Press <PF1><PF3> For Help		

The LatestEd routine is shown below. It finds the row of the Guides Parent Reference and then compares the Revision_Date pointed to by the current value in Latest_Edition_Gui with the Revision_Date sent as an argument. If the Revision_Date sent as an argument is more recent, then it replaces the value in Latest_Edition_Gui with the Row ID of the row in Editions being inserted.

```
LatestEd(Newed,Newdate)      ;Trigger in Editions for Latest_Edition_Gui in Guides
;
SET guidesid=$P(Newed,"||")
&sql(SELECT Latest_Edition_Gui->Revision_Date INTO :curdate FROM Guides WHERE G
uides.Guides=:guidesid)
;
IF SQLCODE'=0 W !,"Error: SQLCODE=",SQLCODE Q
;
IF curdate<Newdate Do Update
;
exit Q
;
Update &sql(UPDATE Guides(Latest_Edition_Gui) VALUES (:Newed) WHERE Guides.Guides
```

Example of SQL Action Type Trigger

This is an example of a SQL action type. In this example, we will use the Employees base table and the Departments base table. The Employees base table contains a field Department, which is a designative reference field into the Departments base table. The Departments base table contains the field

NumOfEmp, which is a counter for the number of employees assigned to each department.

This example shows the definition of three triggers for the base table Employees: a Pre-Filing INSERT trigger, a Pre-Filing UPDATE trigger, and a Pre-Filing DELETE trigger. These triggers are used to automatically increment or decrement the value of the NumOfEmp field in the Department base table whenever a new employee joins the department, an employee leaves the department, or an employee changes departments.

Add a New Employee

When a new employee row is added to the Employees base table, we want to increment the NumOfEmp field in the Departments base table by 1. To do this, we create a Pre-Filing INSERT trigger of Action Type SQL, as shown below:

-----Base Table Definition-----			
Base Table Name		Description	
Employees		Employee Information	
Use Default Physical Structure? Yes_			
< Fields >			
-----Trigger Definition-----			
Action Type	Form/Window/Field Query/Routine	(M Expressions)	
SQL		IF	
		IF	
		IF	
		IF	
-----SQL Lines-----			
SQL Lines (2/2)			
UPDATE Departments SET NumOfEmp=NumOfEmp+1			
WHERE Departments=:%data(6)			
...SQL Lines		Unsaved Data Press <PF1><PF3> For Help	

The WHERE clause identifies which row in the Departments base table should be incremented, namely the Departments row whose Row ID is equal to the value of %data(6), which contains the value of the Department Designative Reference field (Column Number 6) in the new employee row.

Employee Changes Departments

When an employee moves from one department in the company to another, we want to decrement the NumOfEmp field in the department the employee has left and increment NumOfEmp in the department the employee has joined.

To do this, we create a Pre-Filing UPDATE trigger with two trigger items, one to increment the counter, another to decrement the counter. Both trigger items use Action Type SQL, and both trigger items are conditional.

The first trigger item increments the NumOfEmp field in the row of the Departments table for the employee's new department. The row is specified by the M variable %data(6), which is equivalent to the new value of the Designative Reference field Department in the Employees table.

The M IF condition, \$DATA(%data(6)), provides that the trigger item only executes when the M variable %edit(6) has a defined value:

-----Base Table Definition-----		
Base Table Name	Description	
Employees	Employee Information	
Use Default Physical Structure? Yes_		
< Fields >		
-----Trigger Definition-----		
Action Type	Form/Window/Field Query/Routine	(M Expressions)
SQL		IF \$DATA(%data(6))
		IF
		IF
		IF
-----SQL Lines-----		
SQL Lines (2/2)		
UPDATE Departments SET NumOfEmp=NumOfEmp+1		
WHERE Departments=:%data(6)		

...SQL Lines		
Unsaved Data		
Press <PF1><PF3> For Help		

The second trigger item decrements the NumOfEmp field in the row of the Departments table for the employee's old department. The row is specified by the M variable %edit(6), which is equivalent to the old value of the Designative Reference field Department in the Employees table.

The M IF condition, \$DATA(%edit(6)), provides that the trigger item only executes when the M variable %edit(6) has a defined value:

-----Base Table Definition-----		
Base Table Name	Description	
Employees	Employee Information	
Use Default Physical Structure? Yes_		
< Fields >		
-----Trigger Definition-----		
Action Type	Form/Window/Field Query/Routine	(M Expressions)
SQL		IF \$DATA(%edit(6))
SQL		IF \$DATA(%edit(6))
		IF
		IF
-----SQL Lines-----		
SQL Lines (2/2)		
UPDATE Departments SET NumOfEmp=NumOfEmp-1		
WHERE Departments=:%data(6)		
...SQL Lines		
Unsaved Data		
Press <PF1><PF3> For Help		

Employee is Deleted

When an employee leaves the company, we want to decrement the NumOfEmp field in the Departments base table by 1. To do this, we create a Pre-Filing DELETE trigger of Action Type SQL. Note that the SQL command used below is the UPDATE command:

-----Base Table Definition-----		
Base Table Name	Description	
Employees	Employee Information	
Use Default Physical Structure? Yes_		
< Fields >		
-----Trigger Definition-----		
Action Type	Form/Window/Field Query/Routine	(M Expressions)
SQL		IF
		IF
		IF
		IF
-----SQL Lines-----		
SQL Lines (2/2)		
UPDATE Departments SET NumOfEmp=NumOfEmp-1		
WHERE Departments=:%data(6)		
...SQL Lines		
Unsaved Data		
Press <PF1><PF3> For Help		

Chapter 11—Base Table Triggers

This trigger decrements the NumOfEmp field in the row of the Departments table for the employee's old department. The row is specified by the M variable %data(6), which is equivalent to the old value of the Designative Reference field Department in the Employees table.

Examples of Mixed Action Type Triggers

The following are examples of a mixed action type. For these examples, we will use a base table Invoice that has a child table Line_Items.

The Invoice base table has the following fields of interest:

Table 11-6: Invoice Base Table Description

Field Name	Data Type	Column Number
Invoice_Num	RowID	1
Total_Invoice_Amount	Number	2
Last_Transaction_Date	Date	3
Master_Invoice_Num	Designative Reference to another row in the Invoice table, which is the master invoice under which this one is created.	4

The Line_Items child table has the following fields of interest:

Table 11-7: Line_Items Child Table Description

Field Name	Data Type	Column Number
Line_Item	RowID	1
Invoice_Num	Parent Reference	0
Line_Number	Child Subscript	2
Amount	Number	3

The transactions described on the following pages show how the use of base table triggers can enforce database integrity.

Add a Line Item to an Invoice

When a new line item is added, we want to automatically update the Total_Invoice_Amount and Last_Transaction_Date fields in the Invoice base

table. To do this, we add the following sequence of trigger items to the Line_Items child table either Pre-Filing INSERT or Post-Filing INSERT.

Trigger Item 1

This trigger item creates an M local variable "total" and gives it the current value of the Total_Invoice_Amount field in the Invoice base table.

```
Action Type:  SQL
SELECT Total_Invoice_Amount INTO :total FROM Invoice
WHERE Invoice_Num = :%data(0)
```

Trigger Item 2

This trigger item sets the local variable "total" equal to the sum of the existing value of "total" and the value of the Amount field in the Line_Items child table. It also sets the M local variable "today" equal to the current date.

```
Action Type:  M Code
Set total=total+{Amount};  increment invoice total
Set today=$P($H,"",1);  get today's date
```

Trigger Item 3

This trigger item updates two fields in the Invoice base table. It updates the Total_Invoice_Amount field with the value of the local variable "total". It also updates the Last_Transaction_Date field with the value of the local variable "today".

```
Action Type:  SQL
UPDATE Invoice SET Total_Invoice_Amount = :total,
Last_Transaction_Date = :today WHERE Invoice_Num =
:%data(0)
```

Edit a Line Item Amount

When the Amount field of a line item is edited, we want to automatically adjust the Total_Invoice_Amount field as well as update the Last_Transaction_Date field. To do this, we add a Post-Filing UPDATE trigger to the Line_Items child table.

In order to get the delta amount of the transaction, we need to subtract the old line item amount from the new line item amount. For the new amount, we can use the {Amount} designation, but for the old amount, we must reference the value stored in the %edit array, as follows:

```
%edit(column_number)
```

Trigger Item 1

This trigger item creates an M local variable "total" and gives it the current value of the Total_Invoice_Amount field in the Invoice base table.

```
Action Type:  SQL
SELECT Total_Invoice_Amount INTO :total FROM Invoice
WHERE Invoice_Num = :%data(0)
```

Trigger Item 2

This trigger item sets the local variable "total" equal to the existing value of "total" plus or minus the net adjustment of the Amount field in the Line_Items child table. It calculates the net adjustment of the Amount field as the current value of "Amount" minus the old value of "Amount". It also sets the M local variable "today" equal to the current date.

```
Action Type:  M Code
SET total=total+{Amount}-%edit(3);increment invoice total
SET today=$P($H,"",1) ; get today's date
```

Trigger Item 3

This trigger item updates two fields in the Invoice base table. It updates the Total_Invoice_Amount field with the value of the local variable "total". It also updates the Last_Transaction_Date field with the value of the local variable "today".

```
Action Type:  SQL
UPDATE Invoice SET Total_Invoice_Amount = :total,
Last_Transaction_Date = :today WHERE Invoice_Num =
:%data(0)
```

Delete a Master Invoice

Suppose a master invoice has some subsidiary invoices in a one-to-many relationship, but the subsidiary invoices are not really child rows. Instead, the subsidiary invoices are rows in the same table with a designative reference to the master invoice, and there is no link going the other way. The objective is to automatically delete all subsidiary invoices when the master invoice is deleted.

To do this, we add a Pre-Filing DELETE trigger to the Invoice table. This is consistent with the way Open M with SQL handles child table deletions, even though these are technically not child tables.

Trigger Item 1

This trigger item deletes all invoices in the Invoice table where the value of the Master_Invoice_Num Designative Reference field is equal to the RowID of the invoice to be deleted.

Action Type: SQL

```
DELETE FROM Invoice WHERE Master_Invoice_Num = :%data(1)
```

Base Table Help and Error Messages

This chapter describes how to further define a base table in the Data Dictionary. The essentials of base table definition are described in Chapter 5, *Defining a Base Table* .

Specifically, this chapter covers the following topics:

n	Base Table Access Privileges	page 12-2
n	Base Table Advanced Options	page 12-3
n	Help Text	page 12-5
n	Row Insert and Delete Messages	page 12-7
n	Developer Comments	page 12-11

Base Table Access Privileges

Initially, only the user `_SYSTEM` and the owner (creator) of a base table have full privileges for the base table. Users with full privileges have all of the following individual privileges:

- | | | |
|---|------------|--|
| n | %ALTER | Modify and/or delete the base table definition |
| n | SELECT | Use the base table as the data source of a Form Generator form, M/PACT report or SQL statement |
| n | INSERT | Insert rows into the table |
| n | UPDATE | Update existing rows |
| n | DELETE | Delete rows |
| n | REFERENCES | Define designative reference fields |

In the Open M with SQL object security system, a base table is an Open M with SQL object of type **table**. You can control what privileges other users have in relation to the object. Using the GRANT Privileges window, which is accessed from the Privileges Menu, the `_SYSTEM` user and the base table's owner can grant privileges to other users.

See the discussion of object security and the Privileges Menu in the *Open M/SQL Developer Guide* for details.

Base Table Advanced Options

You may use the Advanced Options option on the Base Table Definition horizontal options menu to access the Advanced Base Table Options window.

The Advanced Base Table Options window displays the following information about your base table:

- n Owner (this value is read-only)
- n Internal ID# (this value is read-only)
- n Routine Name (this value is modifiable)

The Advanced Base Table Options window also lets you define the following additional components for your base table:

- n A list of help topics
- n Customized messages (in several international languages) for row insert and row delete
- n A locking reference to be used instead of the full global reference of the master map

Procedure To access and use the Advanced Base Table Options window:

1. From the Base Table Definition main window, press the <GO TO BOTTOM MENU> key to access the Base Table Definition horizontal options menu.
2. From the Base Table Definition horizontal options menu, select the Advanced Options option.

You see the Advanced Base Table Options window, as shown below:

-----Base Table Definition-----

Base Table Name

Description

Guides

Use Default Physical Structure? Yes_

-----Advanced Base Table Options-----

BaseTable Owner: _SYSTEM

Routine Name: mt15

< Help Topics > < Messages > < Notes >

Override Lock Reference:

Enable Field Length Check? Table ID# (irn) 15

...Advanced Options

Unsaved Data

Press <PF1><PF3> For Help

Implicit Joins	Validation Code	BaseTable Triggers	Compile Table	Comp Rel Objects	Copy Field	Advanced Options	Comments
----------------	-----------------	--------------------	---------------	------------------	------------	------------------	----------

The following table lists and describes the fields located on the Advanced Base Table Options window:

Table 12-1: Fields on Advanced Base Table Options window

Field	Description
Base Table Owner	This field displays the UserName of the user who created (and therefore owns) the query. You cannot edit this field.
Routine Name	<p>This field displays the prefix for all routine names generated for the base table when it is compiled.</p> <p>The Data Dictionary assigns a default prefix (such as "mt14" in the above example). You may optionally override the default prefix by replacing it with a string of 1 to 7 alphanumeric characters (first character must be alphabetic) to serve as the override prefix.</p> <p>All routines generated for the base table consist of this prefix followed by a single character (1-9, then A-Z, then a-z), which is appended by the compiler.</p> <p>Specifying your own routine prefix may help you to remember it when attempting to reference the base table directly in M code.</p>
Table ID# (irn)	This field displays the internal identification number of the current base table. It is an output-only field.
<Help Topics>	<p>Press <RETURN> to access the Help Topics window where you may define a list of help topics to be attached to the current base table. When creating forms for this base table you can use these help topics to provide context-sensitive help. You may press the <LIST CHOICES> key to see a lookup box that lists all help topics defined in the current database.</p> <p>For more information about help topics, see the <i>Open M/SQL User Interface Programming Guide</i>.</p>

Table 12-1: Fields on Advanced Base Table Options window (Continued)

Field	Description
<Messages>	<p>Press <RETURN> to access the Messages for Form Generator window where you may define customized messages for row insert and row delete to be used by all forms associated with the current base table.</p> <p>For more information about customizing row insert and delete messages, see the section below entitled “Row Insert and Delete Messages” on page 12-7</p>
<Notes>	<p>This field is no longer operational. Instead, use the Comments horizontal menu option from the Base Table Definition screen, as described later in this chapter.</p>
Override Lock Reference	<p>This field allows you to specify a locking reference other than the full global reference of the master map. This feature is used for locking/unlocking rows both in forms and in SQL queries.</p> <p>At the Override Lock Reference field, you may specify an expression to lock and unlock rows, for example:</p> <p style="text-align: center;">^temp(\$j,"has row")</p> <p>Note: You may only include fields used in the RowID in this lock reference. These include the RowID field itself, or fields that the RowID is based upon. You may not use other fields in the lock reference because, at the time of locking, these fields have not yet been retrieved. If you use quoted values, you must enclose them in double quotes (i.e., “value”, not ‘value’).</p> <p>Warning: If the value of any field in this reference changes after locking, the unlock will fail.</p>
Enable Field Length Check?	<p>Yes/No option. If Yes, Open M with SQL performs validation checking on entered field data, checking the data length against the field length definition. If the entered data exceeds the defined field length, it returns an SQL code 104 or 105 indicating a validation error. If No, Open M with SQL does not perform field length validation checking, and stores field data that exceeds the length defined for the field.</p>

- When you complete your work in the Advanced Base Table Options window, press the <PROCEED> key to save your definitions and exit.

Help Text

Open M with SQL gives you the ability to define help text for a base table.

Procedure To define customized help text at the base table level:

- From the Base Table Definition main window, press the <GO TO BOTTOM MENU> key to access the Base Table Definition horizontal options menu.

- 2. From the Base Table Definition horizontal options menu, select the Advanced Options option.

You see the Advanced Base Table Options window.

- 3. From the Advanced Base Table Options window, press <RETURN> on the <Help Topics> action field.

You see the Help Topics window, as shown below:

-----Help Topics-----

Enter a list of help topics to use on all forms based on this base table.

MainPress <PF1><PF3> For Help

- 4. Specify a help topic you wish to define (up to 30 characters long), and press <RETURN>. You see the “Do you want to define a new Help Topic?” popup window. Answer Yes.

You see the Help Topics Definition window, shown below:

-----Help Topic Definition-----

Generate Index Comments

Topic ID: Employees

Title:

Text:
(0/0)

< Related Topics >< Index Items >

ics to
n this

Help TopicPress <PF1><PF3> For Help

- 5. Type in the help topic title and text. You can supply help text of any length; type continuously, line wraps and additional blank lines are presented.

6. Optionally, tab to the <Related Topics> and/or <Index Items> action fields and press <RETURN>. You see a popup window on which to type the appropriate topics for cross-referencing or indexing. You can specify topic names of up to 30 characters.

Row Insert and Delete Messages

Open M with SQL gives you the ability to define customized messages for the insertion and deletion of database rows. These messages appear on screen when users add or delete rows while running an Open M with SQL form.

Customized row insertion and deletion messages are completely optional. If you choose not to define customized messages, Open M with SQL will display the standard messages. The standard messages are:

When a user adds a row to the database:

```
Is this a new entry?
```

When a user deletes a row from the database:

```
Delete this row?
```

You can override these default messages at the base table level as well as at the form level. When you define customized row insert and delete messages at the base table level, your messages become the default messages for all forms associated with that base table. Any messages defined at the form level override the messages defined at the base table level.

You can also define national language translations for each message. Each message may contain up to 60 characters of text.

Open M with SQL displays the customized insert and delete messages in pop-up boxes of variable width, based on the longest message (or translation) which you have defined. Therefore, in some cases, the box may be wider than the text of a particular message.

Procedure To define customized row insert and delete messages at the base table level:

- 1. From the Base Table Definition main window, press the <GO TO BOTTOM MENU> key to access the Base Table Definition horizontal options menu.
- 2. From the Base Table Definition horizontal options menu, select the Advanced Options option.

You see the Advanced Base Table Options window.

- 3. From the Advanced Base Table Options window, press <RETURN> on the <Messages> action field.

You see the Messages For Form Generator window, as shown below:

-----Messages For Form Generator-----

Row Insertion Message (English)

Is this a new entry?_____

< Translations >

Row Deletion Message (English)

Delete this row?_____

< Translations >

MessagesPress <PF1><PF3> For Help

The table below lists and describes the fields located on the Messages For Form Generator window:

Table 12-2: Fields on Messages For Form Generator window

Field	Description
Row Insertion Message	<p>You may enter a message to be displayed when a form asks for confirmation to insert a new row. The message may consist of up to 60 characters of text.</p> <p>If no message is defined at either the base table level or the form level, Open M with SQL will display the standard message: "Is this a new entry?".</p>

Table 12-2: Fields on Messages For Form Generator window (Continued)

Field	Description
Row Deletion Message	<p>You may enter a message to be displayed when the form asks for confirmation to delete a row. The message may consist of up to 60 characters of text.</p> <p>If no message is defined at either the base table level or the form level, Open M with SQL displays the standard message: "Delete this row?".</p>
<Translations>	<p>Press <RETURN> to specify a translation for the customized row insert or row delete message. You may specify translations for this message into any or all of the fourteen national languages supported by Open M with SQL.</p> <p>For more information about defining translations for customized row insert and delete messages, see the section below entitled "Defining Translations for Customized Row Insert and Delete Messages" on page 12-9.</p>

4. When you finish defining your customized row insert and delete definitions, press the <PROCEED> key to save your definitions and exit the Messages For Form Generator window.

Defining Translations for Customized Row Insert and Delete Messages

You may define national language translations for customized row insert and row delete messages into any or all of the fourteen languages supported by Open M with SQL.

The main message is always the English version, but it is not necessary to supply an English version of the message in order to define translations.

Procedure To define language translations for customized row insert and delete messages:

1. From the Messages For Form Generator window, press <RETURN> on the <Translations> action field corresponding to the appropriate message.

This displays the Form Generator Insertion Message Translations (or Deletion Message Translations) window, as shown below:

-----Messages For Form Generator-----

-----Form Generator Insertion Message Translations-----

Language	Translation
Portugues (Port	Esta \e'\ uma entrada nova?
Deutsch (German	Ist das ein neuer Eintrag?
Español (Spanis	\?\Es una entrada nueva?
Svenska (Swedis	
Français (Frenc	
Dutch (Netherla	Is dit een nieuw record?
Italiano (Itali	Nuovo: vuoi inserirlo?
Norsk (Norwegia	
Dansk (Danish)	
Finska (Finnish	

TranslationsPress <PF1><PF3> For Help

The translation entry window is a multi-row form that lets you enter translations for each of the fourteen national languages supported by Open M with SQL.

As you advance through the form, Open M with SQL fills in the languages in the left-hand Language column in the following order:

- n

Portuguese

n

Deutsch (German)

n

Español (Spanish)

n

Svenska (Swedish)

n

Français (French)

n

Dutch (Netherlands)

n

Italiano (Italian)
- n

Norsk (Norwegian)

n

Dansk (Danish)

n

Finska (Finnish)

n

Hanguil (Korean)

n

Thai

n

Catal' (Catalan)

2.

Press <RETURN> until you see the language for which you want to provide a translation, and enter the translation in the Translation field.

Each translation allows up to 60 characters of text.

3.

When you finish entering translations, press the <PROCEED> key to save your definitions and exit the translation entry window.

Run-Time Logic for Displaying Customized Messages

When a user is running a form in English language mode, Open M with SQL uses the following logic to decide which message to display when inserting or deleting a row:

1. If there is a form-level English message, display it.
2. Otherwise, if there is a base table-level English message, display it.
3. Otherwise, display the standard message in English.

When a user is running a form in foreign language mode, Open M with SQL uses the following logic to decide which message to display when inserting or deleting a row:

1. If there is a form-level translation, display it.
2. Otherwise, if there is a base table-level translation, display it.
3. Otherwise, if there is a form-level English message, display it.
4. Otherwise, if there is a base table-level English message, display it.
5. Otherwise, if there is a translation for the standard message, display it.
6. Otherwise, display the standard message in English.

Developer Comments

The Data Dictionary provides a place for developers to write comments on the base table. This comments page is stored with the base table in the ^mcomment global, and is automatically deleted when the base table is deleted. This same form is used for comments on views.

Note: This feature replaces the “Note” option of the Base Table Advanced Options window.

To access the base table comments, go to the Comment option on the Base Table Definition screen and press <RETURN>. This displays the following window:

```

+-----Developer Comment Entry-----+
| Object Type: Base Table_ Object ID: Accounts_ |
+-----+
+-----+
| Reports |
+-----+
| Date      Time      Open M User Name      Developer's Name      Developer Key |
| 01/27/97  04:51PM  _SYSTEM                _____          _____ |
| Comment   _____ |
| Text      _____ |
|           _____ |
|           _____ |
| Date      Time      Open M User Name      Developer's Name      Developer Key |
| _____ |
| Comment   _____ |
| Text      _____ |
|           _____ |
|           _____ |
+-----+
+-----+
Developer Comment Entries                                     Press <PF1><PF3> For Help
Enter the name of the Developer writing this Comment.

```

The Developer Comment Entry window provides space for extensive comments on changes to the base table. When you tab to a blank comment, the Data Dictionary automatically fills in the date, time, and Open M User Name (login name). The comment text you type automatically wraps to the next line, so you can type continuously, without line returns. The comment section shows three lines of text at a time, but this area scrolls forward to accept comments of greater than three lines.

Subsequent access to this window shows the previous comments in reverse chronological order, with the most recent comment presented first. To add a new comment, tab through the existing comments until you reach a blank comment form. The Data Dictionary will resequence this form to the top of the list the next time the Developer Comment Entry window is accessed.

Comment Reports

If you select the Reports option on the comments window, it displays the Comments Reports pull-down menu, as shown below:

-----+
| Display Comments (From Oldest to Newest) |
| Display Comments (From Newest to Oldest) |
| Search Developer Comments |
| Developer Comments Master Report |
-----+

Display Comments (from Oldest to Newest) displays an output device selection window. Specify the desired output device, then press <RETURN> at the <Proceed> prompt to display (or print) all comments for the current table in chronological order.

Display Comments (from Newest to Oldest) displays an output device selection window. Specify the desired output device, then press <RETURN> at the <Proceed> prompt to display (or print) all comments for the current table in reverse chronological order.

Search Developer Comments allows you to search comments by search string and object type. This searches all comments in the data base. You can limit this search to just base tables, or just views, or any combination of object types that you specify in the Object Type To Search field. Press <List Choices> from this field to pick the desired object types for this search. Search Developer Comments displays an output device selection window. Specify the desired output device, then press <RETURN> at the <Proceed> prompt to display (or print) all comments in the database that meet the search criteria you specified.

Developer Comments Master Report displays the Developer Comments Master Report Selection Criteria window, as shown below:

+-----Developer Comments Master Report Selection Criteria-----+	
Developer Name Contains: _____	
AND Developer Key Contains: _____	
Date Greater Than or Equal To: _____	
AND Date Less Than or Equal To: _____	
+-----+	
Developer Comment Master Rep	
Press <PF1><PF3> For Help	
Enter any kind of text here.	

This option searches all comments in the database for comments that match these search criteria. Specify the desired output device, then press <RETURN> at the <Proceed> prompt to display (or print) all comments in the database that meet the search criteria you specified.

Field Help and Error Messages

If you use Form Generator forms for data entry in your application, you can take advantage of Open M with SQL's context-sensitive help. This chapter describes how to enter context-sensitive help for each field as part of Data Dictionary field definition. The topics covered include:

- n Overview of the Field Help Facility page 13-2
- n Defining Long Help and Error Messages page 13-4
- n Defining Override Help page 13-8

Overview of the Field Help Facility

Open M with SQL allows application developers to create easy-to-use, context-sensitive on-line help about fields if your application uses Form Generator forms for data entry.

M with SQL Help Menu

You see this menu when you press <Help>. The keys which activate Help are shown as the right side of the status line at the bottom of most windows.

Chapter 3, *Using the Data Dictionary Interface*, provides an introduction to this menu. Learn more from the *Open M/SQL Form Generator Guide*.

Three Types of Field Help

You can define three types of messages for each field in your relational database, which are displayed if you use Form Generator forms for data entry:

- n Error messages (which also act as short help messages)
- n Long help messages
- n Override help messages

Error Messages

There are two locations where you can define the contents of a field error message:

- n At the Error Message field on the Error and Help Messages window
- n As the value of %msg if you defined field validation code at the Conversion and Validation Code window. Open M with SQL displays the value you defined for %msg if the user enters invalid data.

Open M with SQL displays the message you entered at the Error Message field if:

- n The value a user enters for a field does not pass field validation and you did not define a value for %msg.
- n A user presses <Explain> and you did not provide a long or override help message.

Users Press <Explain> to Display Help

If a user presses <Explain>, they see Help or Override Help, whichever is active. If you did not enter a Help Message or Override Help, then Open M with SQL displays the error message you entered.

Override Help

Instead of providing a constant Long Help value for each field in your database, you can define an override help array reference. You put the long help message you want displayed in an M array, which can be either local or global. Optionally, you can enter a routine which is called immediately prior to displaying the help to load a context-sensitive message into the M array.

Choosing Between Long Help and Override Help

Override help is useful in the following situations:

- ⁂ If more than a few of your fields share similar or identical long help messages, you may prefer to set up one form of the message in a global and then point to that global in override help.
- ⁂ If your application already has help messages stored in a global.
- ⁂ If you want to provide context-sensitive help depending, for instance, on the value a user enters into a field.

Defining Long Help and Error Messages

You define short and long help and error messages at the same Error and Help Messages window.

Translations

On the Error and Help Messages window, you provide the English version of each message. However, for each message, you can also provide a translation in each of the national languages Open M with SQL supports. The supported national languages are:

- | | |
|----------------------|---------------------|
| n Catalan (Catal') | n Korean (Hangul) |
| n Danish (Dansk) | n Norwegian (Norsk) |
| n Dutch | n Portuguese |
| n Finnish (Finska) | n Spanish (Español) |
| n French (Français) | n Swedish (Svenska) |
| n German (Deutsch) | n Thai |
| n Italian (Italiano) | |

The System Manager defines the national language in which an application presents messages from the M with SQL System Help Menu.

Procedure To define long help or short help/error message and their translations:

1. Enter the Field Definition master window from the Base Table Definition master window.
2. Press <Go to Bottom Menu>.
3. Select the Error/Help Messages option.

You see the Error and Help Messages window:

```
+-----Error and Help Messages-----+
Override Help? No__      <Edit Override Help>

Error Message (English)
Enter a number such as 1234.56_____
                                     < Error Translations >

Short Help Message (English)
_____|_____
                                     < Short Help Translations >

Long Help Message (English) (0/0)
_____|_____
_____|_____
_____|_____
_____|_____
_____|_____
                                     < Long Help Translations >
+-----+

Error and Help MessagesPress <PF1><PF3> For Help
```

The table below describes the fields on this window:

Table 13-1: Fields on the Error and Help Messages Window

Field	Description
Override Help?	Specify Yes or No. A value of No indicates that no Override Help array reference is defined. A value of Yes indicates that an Override help array may be defined, and disables the Long Help fields of this screen.
<Edit Override Help>	Activated if Override Help?=Yes. Press <RETURN> to see the Override Help window, at which you provide an alternate method for your application to obtain Long Help. Specify an override help array reference, the display width of the override help window, and optionally, a user-defined routine which Open M with SQL will call to load the array before it retrieves the help message.
Error Message (English)	Enter an error message of up to 76 characters. Open M with SQL displays this error message during form data entry if an input value does not pass field validation, or if the user presses <Explain> and there is no Short Help for the field on which the cursor is located.
<Error Translations>	Press <RETURN> to see the Field Error Translations window. There enter translations of your error message in any or all of the supported international languages. A translation may be up to 140 characters in length.
Short Help Message (English)	Enter a Short Help message of up to 76 characters.

Table 13-1: Fields on the Error and Help Messages Window

Field	Description
<Short Help Translations>	Press <RETURN> to see the Field Short Help Translations window. There enter translations of your short help message in any or all of the supported international languages. A translation may be up to 140 characters in length.
Long Help Message (English)	Disabled if Override Help?=Yes. If activated, enter a Long Help message of any length.
<Long Help Translations>	Press <RETURN> to see the Field Short Help Translations window. There enter translations of your short help message in any or all of the supported international languages. A translation may be up to 140 characters in length.

4. Press <TAB> at Override Help to bypass that branching field.
5. Enter the English language version of your error message at Error Message (English).
6. If your application needs to display error messages in alternate languages, press <RETURN> at <Error Translations>.

You see the Error Message Translations window:

```

+-----Error and Help Messages-----+
Override Help? No__      <Edit Override Help>
Error Message (English)
Enter a number such as 1234.56_____
Short Help Message (English)_____
                                < Error Translations >
                                _____
                                < Short Help Translations >
+-----Field Error Translations-----+
Language      Translation
Portugues (Portugues)  Entre com um n\u'\mero, tal como 1234.56_____
Deutsch (German)_____ Bitte geben Sie eine Zahl (z.B. 1234.56) ein._____
Español (Spanish)_____ Entre un n\u'\mero (por ejemplo 1234.56)_____
Dutch (Netherlands)___ Dit veld bevat een nummer in het formaat 1234.56_____
Italiano (Italian)_____ Digita un numero come 1234.56_____
+-----+

Field Error Translations                                Press <PF1><PF3> For Help
Saved...

```

7. Enter the translated version of your error message for each supported national language you want your application to provide.
8. Enter the English language version of your Short Help message at Short Help Message (English).

Defining Long Help and Error Messages

9. If your application needs to display Short Help messages in an alternate language, press <RETURN> at <Short Help Translations>.
10. Enter a Long Help message at the field
11. If your application needs to display Long Help messages in an alternate language, press <RETURN> at <Long Help Translations>.
12. Press <PROCEED> to return to the Base Table Definition master window.

Defining Override Help

As described earlier, you can set up Override Help instead of entering a Long Help message.

Procedure To define override help:

- 1. Select the Help/Error Messages horizontal menu option at the Field Definition master window.
- 2. Press <RETURN> at <Edit Override Help>.

You see the Override Help window:

-----Error and Help Messages-----

Override Help? Yes_<Edit Override Help>

Error Message (English)
Enter any kind of text here._____

< Error Translations >

Short Help Message (English)

Long Help Mess

Help Array Reference

M Routine To Load Override Help

Display Width of Override Help Window __

Override Help

Press <PF1><PF3> For Help

The table below describes the fields on this window:

Table 13-2: Fields on Override Help Window

Field	Description
Help Array Reference	Required. Enter the M array reference (local or global) that contains the text lines. At the level one lower than the reference you enter here, you must include a "0" node which contains the number of lines in the message. Then enter sequentially numbered nodes, starting with 1, which contain the lines of the message. If the array does not exist at runtime, the user receives an error message.

Table 13-2: Fields on Override Help Window

Field	Description
M Routine to Load Override Help	Optional. Enter an entry-point to an M routine you have written which loads text into the M array you named in the field Help Array Reference. You can load all help lines in all Help Arrays referenced for all fields at some other juncture in your application, in which case you leave this field blank in all base table field definitions.
Display Width of Override Help Window	Specify a display width, in number of characters, between 1 and 99.

3. Enter the name of the M array which contains the help text at Help Array Reference.
4. Optionally, enter the name of the M routine which loads the help text into the array you named at Help Array Reference.
5. Enter the display width of the override help window.
6. Press <PROCEED> to return to the Field Definition master window.

The override Help Array Reference and M Routine options support the use of curly brace field names for both database fields and form-only fields.

For example, you could specify a Help Array Reference as follows:

```
help( {department} )
```

Or, you could specify an override help routine reference, as follows:

```
depthelp^help( {department} )
```

In order for this usage to be valid, the referenced curly brace fields must either be computed fields or must be present on some window of the form where the Override Help is used.

Views

Open M with SQL provides a View Definition facility that allows you to create views by joining fields from one or more tables linked by implicit joins (characteristic relationships and designative references).

This chapter introduces the concept of views, explains how to use them, and describes how to define them. Specifically, it covers the following topics:

n	What Is a View?	page 14-2
n	How to Use Views	page 14-3
n	Types of Views	page 14-4
n	Creating a View	page 14-5
n	Defining a Table-Based View	page 14-8
n	Defining a Query-Based View	page 14-17
n	View Definition Options Menu	page 14-21
n	Editing a View	page 14-23
n	Deleting a View	page 14-24
n	Changing a View's Starting Table	page 14-25
n	Copying a View	page 14-28

What Is a View?

In a relational database, tables may be **base tables** or **virtual tables**.

A **base table** is an autonomous, named table that is mapped directly to physical storage structures.

A **virtual table** is a named table derived from one or more base tables. Virtual tables are not directly represented in physical storage. Rather, they are abstract collections of base tables. Examples of virtual tables include query output and **views**.

Views are conceptual windows through which data from one or more base tables can be "viewed". The base tables must be linked via an implicit join: either a designative reference or a characteristic relationship. Views can serve as data sources for M/PACT reports, SQL queries, and other views. They cannot be used as a data source for a form in the Form Generator.

How to Use Views

Within the context of your database, views serve two primary purposes:

1. They are conceptual windows through which a wide range of data from multiple base tables can be simultaneously "viewed". This allows you to pull data across base table boundaries into a single source.
2. They are a security feature that can allow the database designer to restrict access to selected fields and their data.

When you define a view, Open M with SQL stores it in the view table, where it is available for reuse at an time. In this way, views free you from the inconvenience of having to respecify join conditions repeatedly.

You can use views in the following places:

1. Data source of an M/PACT report
2. FROM clause of an SQL query
3. Starting Table of another view

If a view is the starting view of another view, it is called a sub-view. The primary purpose of a sub-view is to restrict access to certain fields of the source view. You cannot define a view based on more than one other view, and you cannot define further views from a sub-view.

Types of Views

You can define two types of views:

- n Table-based Views
- n Query-based Views

Table-based views can be further split into two sub-categories:

- n Views based on base table(s)
- n Views based on another view (sub-views)

All Views Based on SELECT Queries

All views are based on SQL SELECT statement queries.

A table-based view is based on an SQL SELECT statement query that is generated by Open M with SQL from the join specifications and fields to be included information you provide.

A query-based view is based on a directly inserted SQL SELECT statement. The View Definition facility lets you directly insert an SQL SELECT statement query to create query-based view.

Creating a View

Open M with SQL provides two ways to create a view:

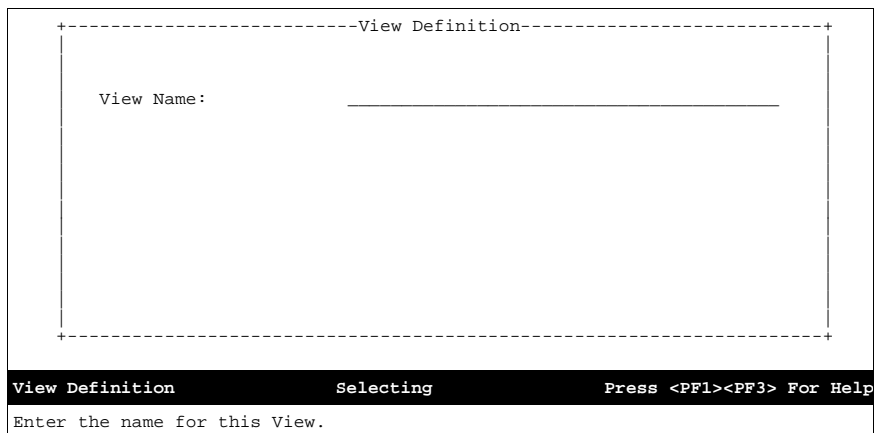
1. Use the View Definition form.
2. Execute a CREATE VIEW DDL statement in the Interactive Query Editor or as embedded SQL in M macro source code.

The latter way creates only query-based views and can be used only as SQL embedded in M macro source code.

Procedure To create a view in the View Definition form:

1. Enter the Data Dictionary
2. Select the View Definition option from the Data Dictionary Menu

You see the View Definition window, as shown below:



3. Enter a name for the view at the View Name prompt, and press <RETURN>.

Note: The view name you enter must not conflict with the name of an already existing view or base table. The view name must follow all of the same naming conventions as a base table.

You will see the "Is this a New View?" popup window.

4. Press <RETURN> to accept the default value Yes.

You see the View Definition master window, as shown below:

View Definition-----

Advanced Options
Copy Query
Reports
Comments

View Name:

Guide_Orders_____

View Description:

This View is Based On:

Query

Join Specified Views must have a Starting Table.
Leave the Starting Table empty if View is Query Based.

Starting Table: _____

<Edit>

< Join Specifications >

< Query >

View Definition
Press <PF1><PF3> For Help

Enter a 1-40 character description of this View.

The following table lists and describes the fields located on the View Definition master window:

Table 14-1: View Definition Master window

Field Name	Description
View Name	<p>This is a required field.</p> <p>Because a view is a table, view names follow the same conventions as base table names. These include a maximum length of 40 characters, which can contain letters, numbers and the underscore "_" character but not other punctuation symbols or blank spaces. A view name must begin with a letter.</p> <p>The view name must be unique within the M database (cannot duplicate another view or base table name) after converted to uppercase and stripped of punctuation. Also, it must not match any SQL reserved word, a list of which is printed in Appendix C, <i>SQL Reserved Words</i>.</p>
View Description	<p>This is an optional field.</p> <p>Enter text that describes the view you are defining.</p>
This View Is Based On...	<p>This is a read-only field that shows what kind of view you are defining. It has three possible values: Base Table, View, and Query.</p>

Table 14-1: View Definition Master window

Field Name	Description
Starting Table	Enter the name of an existing table on which to base this view definition. The starting table can be a base table or another view. If it is a base table, you must deescribe the paths that link together all the other tables on this view. If it is a view, you do not specify any links. Leave this field empty if you want to define a query-based view.
<Edit>	This is an optional field, and is only activated if you specified a starting table. Press <RETURN> on this action field to edit the definition of the starting table.
<Join Specifica- tions>	Press <RETURN> on this action field to invoke the View Join Specification window where you can designate the base tables to be included in this view and how they should be joined.
<Query>	Press <RETURN> on this action field to invoke the Query window, where you can define an SQL SELECT statement query to serve as the foundation of your view.

Defining a Table-Based View

A table-based view is a view based either on a base table or another view.

In either case, to define a table-based view you must first enter a value for the Starting Table field on the View Definition master window. If you define the starting table to be a base table, you create a base table-based view. If you define the starting table to be another view, you create a view-based view.

Starting Table Is the Table on Which View is Based

The Starting Table field designates the table on which the view is based. The starting table can be a base table or another view. Each row of the starting table corresponds to a row of the view.

Think of the starting table as the place where you stand to look out at your database. The starting table is typically a table with no parent. Usually, it has characteristic relationships with and designative references to other tables.

Designative References

When you are working with base tables linked by designative references, keep in mind that **data paths between designative references cannot be traversed backwards**. As such, you should always name the designating base table as the starting table. If you name the referenced table as the starting table, you cannot include fields from the designating table in the view.

Characteristic Relationships

When working with base tables linked by characteristic relationships, you can name either the parent table or the child table as the starting table. If you name the parent table as the starting table, the view has a row for every row of the child table. If you name the child table as the starting table, parent rows that have no corresponding rows in the child table are not included in the view.

Join Specifications Link Base Tables

You use join specifications on base-table-based views only to designate which base tables to include in the view.

In order to include any fields from the starting table in the view, you must explicitly specify the starting table in the Base Table field of the View Join Specifications window.

All other base tables named here must be linked to the starting table by either a characteristic relationship or a designative reference. However, the link does not have to be direct. In other words, the starting table may be linked by a characteristic relationship to Base Table B, and, in turn, Base Table B may be linked to

Base Table C by a designative reference. In this case, you can include Base Table C in the view, provided Base Table B is also included.

Implicit Joins Link Base Tables to the Starting Table

For each base table you add to the view, you must define the join path that links the specified base table to the starting table. You provide this information in the Implicit Join field on the View Join Specification window.

An implicit join designates the path by which data should be retrieved from the linked base tables. The join enables access to data from both the starting and linked tables.

You do not need to define your own joins. Instead, you may press the <List Choices> key to call a lookup that lists every possible path the join can take. Join paths are defined by means of Extended Arrow Syntax.

Implicit Joins Use Extended Arrow Syntax

Extended Arrow Syntax is the mechanism used to syntactically specify links between base tables. It is based on the standard SQL syntax `TableName.FieldName`, with the addition of the arrow syntax, where each arrow in a given statement represents an implicit join.

In join specifications for a view, arrow syntax is used to make the following links:

- n Designative references, as in:
 Editions->Docstaff
 where the "Editions" base table has a designative reference into the "Docstaff" base table.
- n Child-to-parent references, as in:
 Editions->Guides
 where the "Editions" base table is a child of the "Guides" base table.
- n Parent-to-child references, as in:
 Guides->Editions
 where the "Guides" base table is the parent of the "Editions" base table.

Join specifications for views always represent the first link of a join path with a period "." and all subsequent links with an arrow "->".

You can see a lookup box of possible Join Specifications after entering a table name and then pressing <List Choices>.

Examples of Join Specifications

- Example** Our sample application includes the following relationships:
- n Base table "Guides" has a child table "Editions".
 - n "Editions" has a field "Main_Author", which is a designative reference into base table "Docstaff".
 - n The View named GuideView has the starting table "Guides".

The Join Specifications screen is shown below. It shows the join specifications for the first, starting table; we want to select a join path to the second table. We have two choices, since Editions is both a child table and a designated table of Guides.

Base Table Name	Description
Editions	Table of Latest Editions
Editionsc	Editions of Guides

Join Desc:

Implicit Join: Guides

<Fields>

Base Table: Editions

<Edit>

Join Desc:

Implicit Join:

<Fields>

Base Table:

<Edit>

Join Desc:

Implicit Join:

<Fields>

Base Table:

<Edit>

Join Desc:

Implicit Join:

<Fields>

View Join Specification

Press <PF1><PF3> For Help

Enter the name of a Base Table. Press <CHOICE> for a list of Tables.

- Example** Our sample application includes the view Guide_Orders, where we report on orders by Guide Title, Version and Revision_Date. The starting table is

Copy_Orders. We use the designative reference field, Edition_Ordered, to get to the base tables which include some of the fields we need.

View Name: Guide_Orders		
View Join Specification		
Base Table:	Copy_Orders	<Edit>
Join Desc:		
Implicit Join:	Copy_Orders	<Fields>
Base Table:	Guides	<Edit>
Join Desc:		
Implicit Join:	Copy_Orders.Edition_Ordered->Guides	<Fields>
Base Table:	Editions	<Edit>
Join Desc:		
Implicit Join:	Copy_Orders.Edition_Ordered->Guides->Editions	<Fields>
Base Table:		<Edit>
Join Desc:		
Implicit Join:		<Fields>

View Join Specification
Press <PF1><PF3> For Help

Sub-Views Copy Join Specifications Automatically

If you define another view as the starting table of your view, Open M with SQL automatically copies the join specifications defined for the source view. You cannot access the join specifications and, thus, do not have the ability to alter or even view these joins.

The two views are permanently linked. Any modifications you make to the join specifications of the source view are echoed to the sub-view.

The only aspect of the sub-view that is independent of the source view is the list of fields to be included. The master list of fields available to the sub-view is the union list of all fields included in the source view. From this master list, you may select which fields to include in the sub-view. The choices you make do not have any effect on the source view.

You Must Designate Fields to Appear in the View

For each base table that you include in a view, you must define which fields you want to make available.

The Fields on View Definition window lets you designate which fields to include. It also lets you specify an alternate field name for the view, or change the field description.

Rename Fields on View To Avoid Duplication

When you designate the fields that you want to be included on the view, Open M with SQL lets you rename those fields to give them view-specific names and descriptions. These names appear as default column titles in M/PACT reports.

This is an optional feature, unless two or more fields included in the same view share the same field name, in which case you must rename one of them. This can happen when you link separate tables because there is no constraint to prevent different tables from having identical field names.

For example, suppose a field called "Total" appears twice or more in separate base tables that you are linking in the view. If this occurs, the view facility will force you to rename all but the first occurrence of the field "Total" that you choose to include in the view.

Note: The new Field Name that you supply for the view is for the view's use only. It does not replace the old Field Name in the original table.

Example The fields available from the Copy_Orders table are shown on the screen below. This is the first table we list in the Join Specifications for our Guide_Orders view:

View Name: Guide_Orders				
Base Table Name: Guides				
Fields on View Definition				
Table's Field Name	Inc?	View Field Name	Description	
Guides	No			<Edit>
Product	No			<Edit>
Date_Retired	No			<Edit>
Num_Editions	No			<Edit>
Retired	No			<Edit>
Title	No			<Edit>
Who_retired	No			<Edit>
				<Edit>
				<Edit>
				<Edit>
				<Edit>
				<Edit>
				<Edit>

Fields on View Definition Press <PF1><PF3> For Help

Enter 'Yes' to include this field in the view, else enter 'No'.

Procedure To define a table-based view:

1. On the View Definition master window, specify an existing table (base table or view) in the Starting Table field.

You may press the <List Choices> key to see a lookup box that lists all existing tables defined in your database. Use the cursor keys to navigate and press <RETURN> to select the appropriate entry from the list.

Note: Selection of a starting table is an irreversible decision. Once you specify a table, the Starting Table field becomes inaccessible, preventing you from changing its value. A Data Dictionary utility called Change a View's Starting Table lets you do this. See the section later in this chapter entitled "Changing a View's Starting Table" on page 14-25 for information on how to use this utility.

2. Press <RETURN> on the <Join Specifications> action field in the View Definition master window.

If you are defining a view that is based on another view, you do not see the View Join Specification window. Open M with SQL automatically copies the join specifications defined for the source view. Skip ahead to Step 6.

If you are defining a base table-based view, you see the View Join Specification window, as shown below:

View Name: Guide_Orders		
-----View Join Specification-----		
Base Table:	_____	<Edit>
Join Desc:	_____	
Implicit Join:	_____	<Fields>
Base Table:	_____	<Edit>
Join Desc:	_____	
Implicit Join:	_____	<Fields>
Base Table:	_____	<Edit>
Join Desc:	_____	
Implicit Join:	_____	<Fields>
Base Table:	_____	<Edit>
Join Desc:	_____	
Implicit Join:	_____	<Fields>
View Join Specification Press <PF1><PF3> For Help		
Enter the name of a Base Table. Press <CHOICE> for a list of Tables.		

The following table lists and describes the fields located on the View Join Specification window:

Table 14-2: View Join Specification Window

Field Name	Description
Base Table	This field takes the names of base tables you want to include in the view. You must include the starting table as one of the base tables in the view.
<Edit>	Press <RETURN> on this action field to edit the definition of the base table.
Join Description	This field is optional. You may enter a description of how the table can be used in this view. It is also useful to fully describe the join path and how it relates to the starting table.
Implicit Join	Enter the path that links the starting table to this base table. This path is actually a series of join conditions that link designative references and characteristic relations. You can press <List Choices> to see the possible paths.
<Fields>	Press <RETURN> on this action field to invoke the Fields on View Definition window where you can designate the fields that should be included in the view.

3. Enter the names of the base tables you want to include in the view in the Base Table field.

The Base Table field does not support an unqualified lookup on all base tables in the database. It does, however, support lookups on partial information. You must know the first few characters of the names of all base tables that you want to include on your view.

4. Enter the join path that links each base table to the starting table in the Implicit Join field.

You do not need to define your own join paths in this field. Instead, you may press the <List Choices> key to automatically fill in the join path. If there is more than one path the join can take, this calls a lookup box that lists every possible path.

5. For each base table, press <RETURN> on the action field <Fields>.

6. Type Yes at the Inc? (Include In View) field for each field that you want to include in the view.
7. Give each field included in the view a view-specific Field Name and Field Description.

This is an optional step.

The default Field Name and Field Description is the same as defined for the base table. You may keep these values, or you may override them.

8. Optionally, edit the original field definition using the <Edit> action field.
9. When you complete the join specifications and fields to be included, press the <PROCEED> key to save your work and exit the View Definition window.

3. In the Query window, enter an SQL SELECT statement query to define the view.
4. When you complete your query definition, press the <PROCEED> key to save your work and exit the View Definition window.

Copy an Existing Query Into the Current View

Open M with SQL provides a special facility that lets you copy an existing query into the Query window of your view. This includes any existing query defined using either the Query Generator or Query Editor.

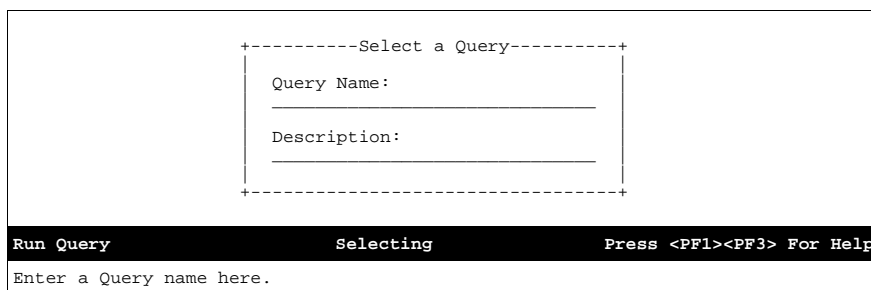
You can only use this facility when the following conditions are true:

1. The view is query-based.
2. No query is yet defined for the view.

Procedure To copy an existing query into the current view:

1. On the View Definition master window, press the <Go To Bottom Menu> key to access the View Definition horizontal options menu.
2. On the View Definition horizontal options menu, select the “Copy Query” option.
3. From the displayed list, select "Copy a Query Into This View".

You see the Select a Query window, as shown below:



4. Enter the name (and optionally the description) of the existing query you want to copy, and press <RETURN>.

You may press the <List Choices> key to see a lookup box that lists all existing queries. Use the cursor keys to navigate and press <RETURN> to select the appropriate entry from the list.

5. Press the <PROCEED> key to initiate the copy.

The copied query appears in the Query window.

Copy Another View's Query Into the Current View

Open M with SQL provides a special facility that lets you copy the query of another view into the Query window of the current view.

You can only use this facility when the following conditions are true:

1. The current view is query-based.
2. No query is yet defined for the current view.

The view whose query you are copying does not have to be a query-based view; it must only have a query defined in its Query window.

Procedure To copy the query of an existing view into the current view:

1. On the View Definition master window, press the <Go To Bottom Menu> key to access the View Definition horizontal options menu.
2. On the View Definition horizontal options menu, select the "Copy Query" option.
3. From the displayed list, select "Copy Another View's Query Into This View".

You see the Select a View window, as shown below:

-----Select a View-----

View Name:

View Selection

Selecting

Press <PF1><PF3> For Help

Enter the name for this View.

4. Enter the name of the existing view whose query you want to copy, and press `<RETURN>`.

You may press the <List Choices> key to see a lookup box that lists all existing views. Use the cursor keys to navigate and press <RETURN> to select the appropriate entry from the list.

- 5.** Press the <PROCEED> key to initiate the copy.

The copied query appears in the Query window of the current view.

You Can Convert a Query-Based View to Table-Based

If you originally define the view as query-based, you may later return to the View Definition window and enter a value (an existing base table or view) in the Starting Table field to convert the view to table-based. This, however, is irreversible. Once you have defined it to be table-based, you cannot later convert back to query-based. In this case, the defunct query remains available read-only in the Query window. It is also available to be copied into another view.

You Cannot Convert a Table-Based View to Query-Based

Entering a value into the Starting Table field on the View Definition window to define a view as table-based is an irreversible step. You cannot later convert a table-based view to query-based.

View-Related DDL Statements for Query-based Views

InterSystems supports the following SQL DDL statements for use with query-based views:

- n CREATE VIEW <viewname> AS SELECT...
- n ALTER VIEW <viewname> AS SELECT
- n DROP VIEW <viewname>

Open M with SQL supports the use of these SQL statements in two places:

1. In SQL code embedded in M macro source code.
2. In the Interactive Query Editor.

ALTER VIEW has exactly the same syntax as CREATE VIEW. It requires you to recreate the entire view but lets you retain the same name and internal view number.

Example Suppose we wanted to provide a partial view of the Guides table, which listed all titles and whether or not they were retired. Instead of creating this view in the View Definition form, we could issue the following SQL statement:

```
CREATE VIEW Guide_History AS
SELECT Guides, Title, Retired, Date_Retired
FROM Guides
```


-----View Definition-----
Advanced Options Copy Query Reports Comments

This option shows you the owner of the view, the ID number assigned to the view, and the date and time that the view was last edited. This option displays the Advanced Options popup window, as shown below:

The owner is the UserName of the user who created the view. The owner can grant and revoke privileges on the view in the Open M with SQL RDBMS security system.

Open M with SQL Data Dictionary Guide 14-21

View Definition Reports

This option displays the Reports pull-down menu, as shown below:

View Definition	
Advanced Options	Copy Query Reports Comments
View Name: Acct	List of Reports Based on this View
View Description: _____	View Definition - Sorted by Path
This View is Based On: Base	View Definition - Sorted by Fields
<div>Join Specified Views must have a Starting Table. Leave the Starting Table empty if View is Query Based.</div>	
Starting Table: Accounts _____ <Edit>	
< Join Specifications > < Query >	

View Definition Reports 04:44PM Press <PF1><PF3> For Help

If you select the List of Reports Based on this View option, you get the Reports with This View as a Data Source window. This window lists the reports by name with their description and a Report ID number. Each listed report has an <Edit> option which allows you to edit the report by going to the Report Definition window.

For further details on the other two report options, see “View -- Sorted by Path Report” on page 15-9 and “View -- Sorted by Fields Report” on page 15-11 in Chapter 15, *Data Dictionary Reports* .

View Definition Comments

This option permits you to record developer comments for the current view. For further details, see “Developer Comments” on page 12-11 in Chapter 12, *Base Table Help and Error Messages* .

Editing a View

Once you have created a view, you may later reenter the View Definition window for that view and edit it.

Any user with %ALTER privileges for a specific view can edit that view. The owner of the view automatically has %ALTER privileges. Along with the Open M with SQL System Manager, the owner can give other users %ALTER privileges to any owned base tables, using the Open M with SQL Privileges menu option.

When you edit a view, you can edit the following elements of its definition:

- n The view name
- n The view description
- n The query (if it is a query-based view)
- n The join specifications
- n The fields to be included from each of the linked base tables
- n The field name for each of the fields included in the view
- n The field description for each of the fields included in the view

You cannot edit the view's starting table. To change a view's starting table, you must use a separate utility described in the section “Changing a View's Starting Table” on page 14-25.

Procedure To edit a view:

1. Enter the Data Dictionary.
2. Select the View Definition option from the Data Dictionary menu.
3. At the View Definition row selection window, enter the name of the view you wish to edit and press <RETURN>, or press the <Search Current Table> key from a blank View Name field to see a lookup box that lists all existing views.

Alternatively, you may partially enter the view name and press the <Search Current Table> key to see a lookup box that lists all views that match the partial information.

4. In the lookup box, use the <Up Arrow> and <Down Arrow> keys to position the highlight bar on the desired entry, and press <RETURN> to select it.

You see the View Definition master window.

5. Edit the view definition.

6. When you have finished editing the view, press the <PROCEED> key to save the view and exit the View Definition form.
7. Edit any reports or queries affected by the changes made to this view.
8. Recompile all reports based on this view.

See the *Open M/SQL M/PACT Guide* to learn how to compile reports.

Deleting a View

Any user with the %ALTER privilege for a specific view can delete that view. The owner of the view automatically has %ALTER privileges. Along with the Open M with SQL System Manager, the owner can give other users %ALTER privileges to any owned base tables, using the Open M with SQL Privileges menu option.

Deleting a View Automatically Deletes Dependent Objects

When you delete a view, Open M with SQL automatically deletes any queries and reports that reference the view. If the deleted view is the starting table of another view, Open M with SQL automatically deletes the other view.

Procedure To delete a view:

1. Enter the Data Dictionary.
2. Select the View Definition option from the Data Dictionary Menu.
3. At the View Definition row selection window, enter the name of the view you wish to delete and press <RETURN>, or press the <Search Current Table> key from a blank View Name field to see a lookup box that lists all existing views.

Alternatively, you may partially enter the view name and press the <Search Current Table> key to see a lookup box that lists all views that match the partial information.

4. In the lookup box, use the <Up Arrow> and <Down Arrow> keys to position the highlight bar on the desired entry, and press <RETURN> to select it.

You see the View Definition master window.

5. Press the <DELETE ROW> key.
6. Press <RETURN> to accept the Yes default at the "Delete this row?" prompt on the message line.

You see the message "Deleting..." on the message line.

Changing a View's Starting Table

When you specify the starting table for a view in the View Definition window, the Starting Table field immediately becomes inaccessible, thus preventing you from changing its value.

To change the starting table of an existing view, you must use the Change a View's Starting Table utility.

This utility performs the following actions:

- n It replaces the old starting table of a view with a new starting table.
- n It automatically inserts an initial reference to the new starting table in the join specifications for the view.
- n It automatically updates all the old join specifications accordingly.
- n It lets you specify which fields from the new starting table you want to include in the view.

Restrictions on the Use of This Utility

The Change a View's Starting Table utility acts only on base table-based views. It does not act on either view-based views or query-based views.

The only base tables eligible to be named as the new starting table are those that have at least one designative reference field into the current starting table. If you select a table that does not have a designative reference field into the current starting table, you will see the following error message:

```
This table does not designate the starting table of the view.
```

Procedure To change the starting table of a view:

1. Select the Change View's Starting Table option from the Data Dictionary menu.

You see the Change Starting Table for a View window, as shown below:

```

+-----Change Starting table for a view-----+
|
|  View Name                               ViewDesc
|  _____
|
|  Table Name                             Description
|  _____
|
|                                     < Fields >
|
|  Change Starting table of view? ____
|
+-----+
| Change Starting table for a view          Press <PF1><PF3> For Help |
+-----+

```

2. At the View Name field, enter the name of the view whose starting table you want to change.

You may press the <List Choices> key to see a lookup box that lists all existing views. Use the cursor keys to navigate and press <RETURN> to select the appropriate entry from the list.

3. At the Table Name field, enter the name of the base table that you want to appoint as the new starting table of this view.

Note: The only valid base tables are those that have a designative reference into the current starting table.

4. Press <RETURN> on the <Fields> action field to specify the fields from the new starting table that you want to include in the view.

You see the Fields to Include in View window, as shown below:

```

+-----Change Starting table for a view-----+
|
| View Name          ViewDesc
| Guide_Orders_____
|
| Table Name          Description
+-----Fields to include in view-----+
|
|           ----- View Data -----
|           Include
|           in
| Field Name      view? field name      field description
|
| Copy_Orders      No__ Copy_Orders_____ Copy_Orders Row ID_____
| Edition_Ordered  No__ Edition_Ordered_____ Edition_Ordered_____
|
| _____
| _____
| _____
| _____
|
+-----+
|
| ...Fields to include in view                      Press <PF1><PF3> For Help
|

```

The following tables lists and describes the fields on the Fields to Include in View Definition window:

Table 14-4: Fields to Include in View Definition Window

Field Name	Data Type	Description
Field Name	Text	This is an output-only field. It displays the name of every field in the new starting table.
Include In View?	Yes/No	Type <i>Yes</i> to include this field on the view and therefore make it accessible to reports and queries using this view. Type <i>No</i> to exclude it. The default scheme is based on the data type of the field. Designative Reference and Row ID fields default to <i>No</i> . All other fields default to <i>Yes</i> .
Field Name	Text	The default for this field is the field name defined in the base table. You can specify an override field name for the view. This is necessary when the same Field Name appears more than once in the view.
Description	Text	The default for this field is the field description defined for the base table. You can specify an override field description for the view. The override Field Description is used as the default for Report Column Titles in M/PACT when the view is the report data source.

5. Press the <PROCEED> key to save your fields to include definitions and exit back to the Change Starting Table of View window.

6. At the Change Starting Table of View? field, answer *Yes* and press the <PROCEED> key.

This initiates the change and exits you back to the Data Dictionary menu.

Copying a View

Open M with SQL provides a utility that lets you automatically create a new view by copying an existing view. This utility produces a copied view that is virtually identical to the source view. You might use this utility if you need a view which follows the same Join Specifications, but includes additional fields for a different group of users.

You Must Have Privileges to Copy a View

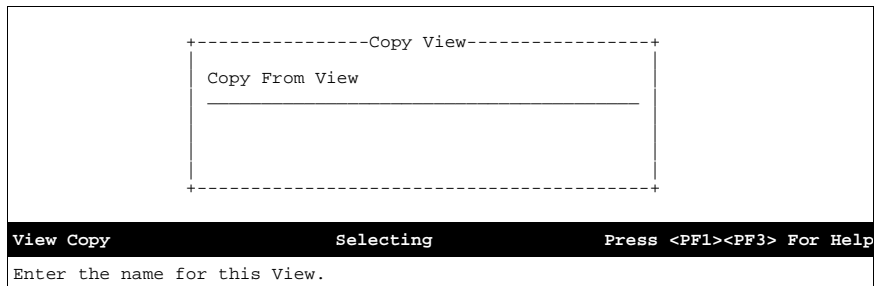
In order to copy a view, you must have the following privileges:

- n %ALTER privileges on the source view.
- n SELECT privileges on each of the base tables comprising the source view.

Procedure To copy a view:

1. Select the Copy a View option from the Data Dictionary menu.

You see the Copy View window, as shown below:



2. At the Copy From View field, enter the name of an existing view.

You may press the <Search Current Table> key to see a lookup box that lists all existing views. Use the cursor keys to navigate and press <RETURN> to select the appropriate entry from the list.

When you select the source view, you see the Copy From View master window, as shown below:

<div style="border: 1px dashed black; padding: 10px; margin: 10px auto; width: 60%;"> <div style="text-align: center; border-bottom: 1px dashed black; margin-bottom: 5px;">--Copy View--</div> <div>Copy From View</div> <div>Guide_Orders</div> <div>Copy To View</div> <div style="border-bottom: 1px solid black; margin-top: 5px;"></div> </div>	
View Copy	Press <PF1><PF3> For Help
Enter the name of the View to copy to.	

3. At the Copy To View field, enter the name of the new view to be created.

Note: This name must not be in use already.

4. Press <PROCEED> to initiate the copy.

Open M with SQL displays a success message when the copy is complete, then returns you to the Data Dictionary menu.

Data Dictionary Reports

Open M with SQL supplies a set of formatted Data Dictionary reports. These reports allow you to display or print out the base table, field, and view definitions in the Data Dictionary and the globals to which they are mapped.

This chapter begins with an explanation of how to run a Data Dictionary report, then discusses each of the reports individually. This chapter includes:

n	How To Run a Data Dictionary Report	page 15-2
n	List of Base Tables Report	page 15-4
n	Base Table (Order by Field Name) Report	page 15-5
n	Base Table (Order by Column Number) Report	page 15-6
n	Series of Base Tables Report	page 15-7
n	List Table Relationships Report	page 15-8
n	List of Views Report	page 15-9
n	View -- Sorted by Path Report	page 15-9
n	View -- Sorted by Fields Report	page 15-11
n	Global Documentation Report	page 15-13
n	Global Doc -- Map Version Report	page 15-15

How To Run a Data Dictionary Report

The following procedure describes how to run a Data Dictionary report. All the reports follow this general model, though some have additional requirements which are described in the section that describes the individual report.

Procedure To run a Data Dictionary report:

1. From the Data Dictionary main menu, select the Reports on Data Dictionary option.

You see the Reports on Data Dictionary window, as shown below:

```

----- OPEN M Developer -----
+-----Reports on Data Dictionary-----+
|
|  List of Base Tables
|  Base Table (Order By Field Name)
|  Base Table (Order By Column Number)
|  Series of Base Tables
|  List Table Relationships
|  List of Views
|  View - Sorted by Path
|  View - Sorted by Fields
|  Global Documentation
|  Global Doc - Map Version
|
+-----+

Thursday Jan 02, 1997                      Directory: /us/susang/
                                           Copyright (c) 1993 - InterSystems Corporation

Reports on Data Dictionary  02:02PM                      Press <PF1><PF3> For Help
Press <RETURN> for a list of all base tables in this database.

```

2. From the Reports on Data Dictionary menu, select the report you want to run.

You see the Device Selection window, as shown below:

```

+-----Device Selection-----+
|
|  Device
|  /dev/ttypl_____
|
|  Description
|  BJB-Jan 2
|
|  Print Format
|  _____
|
+-----+

Device Selection                      Press <PF1><PF3> For Help
Exit Without selecting

```

The following table lists and describes the fields located on the Device Selection window:

Table 15-1: Fields on Device Selection Window

Field	Description
Device	This field always defaults to your current device (the current value of %IS). You may change the default to any valid output device to which your device is linked.
Description	This field reflects the description given to the specified device in the definition table.
Print Format	Here you may select any print format defined for the specified device.

3. In the Device field, enter the name of the device to which you want to send this report.

You can send the report to any valid output device to which your current device is linked.

The default device name is your current device.

To send the report to your screen, press the <PROCEED> key.

To send the report to another device (such as a printer), delete the name of the default device at the Device prompt, enter the name of the new target device, adjust the Print Format parameter as appropriate, and press the <PROCEED> key.

Note: When output to the screen, the report displays one screen at a time and prompts you to press <RETURN> to scroll ahead to the next screen.

List of Base Tables Report

The List of Base Tables Report lists the names and descriptions of all base tables defined in the current directory. The list is ordered alphabetically by the name of the base table.

Below is a sample List of Base Tables Report:

Base Table Name	Description
Accounts	Customer Accounts Master Table
Copy_Orders	Documentation print orders
Docstaff	Information on documentation staff
Editions	Table of Latest Editions
Editionsc	Editions of Guides
Employees	Employee Information
FavFlavors	People's Favorite Flavors
Guides	List of all revisions to guides
Invoices	Invoices for Accounts
People	People in ^FLAVORS global
Press <Return> to Exit	

Base Table (Order by Field Name) Report

The Base Table (Order by Field Name) Report displays a list of all fields defined for a selected base table in alphabetical order by field name. It provides basic information about the selected base table and detailed information about each of its fields.

When you run this report, you first see the Base Table Selection window, as shown below:

-----Base Table Selection-----		
Base Table Name: _____		

Report on Base Table	Selecting	Press <PF1><PF3> For Help
Enter a unique Base Table name, without spaces or punctuation except "_".		

You must specify the name of the base table for which you want to run the report in the Base Table Name field.

You may press the <List Choices> key to see a lookup box that lists all base tables defined in the current directory.

Below is a sample Base Table (Order by Field Name) Report:

```

-----
Definition of Base Table: Accounts    (accounts base table) Approximately 250 Rows

Parent Table:
Child Table(s): Invoices
Master Map Global Reference: ^ACCT("ACCOUNTS",0,{Accounts})
Row ID Field Name: Accounts    Base on Other Fields? No
Based on Fields:
Generated Routine: mt21
-----

```

Field Name	Description	Data Type	Max. Len.	Uniq.	Col No.	Reg.	Multi Line
Accounts	Accounts Row ID	Row ID	16	Yes	1	Yes	No
Account_Name	Name of Account	Text	24		6	No	No
Account_num	account number	Number	10		2	No	No
City_Acct	City of account	Text	24		3	No	No
Name_Acct	Name of Account Holder	Name	32		4	No	No
Phone_Acct	Phone Number	Number	10		5	No	No

Press <Return> to continue, <Options> to scroll, <Exit> to Exit

For each field in the selected base table, it provides the following detailed information (reading left to right):

- n Field Name
- n Field Description
- n Data Type
- n Maximum Field Length
- n Is the Field Unique?
- n Column Number
- n Is the Field Required?
- n Is it a Multi-line Field?

Base Table (Order by Column Number) Report

The Base Table (Order by Column Number) Report displays a list of all fields defined for a selected base table in column number order. It provides basic information about the selected base table and detailed information about each of its fields.

When you run this report, you first see the Base Table Selection window, as shown below:

-----Base Table Selection-----

Base Table Name: _____

Report on Base Table

Selecting

Press <PF1><PF3> For Help

Enter a unique Base Table name, without spaces or punctuation except "_".

You must specify the name of the base table for which you want to run the report in the Base Table Name field.

You may press the <List Choices> key to see a lookup box that lists all base tables defined in the current directory.


```

-----
Definition of Base Table: Accounts      (accounts base table) Approximately 250 Rows

  Parent Table:
  Child Table(s): Invoices
  Master Map Global Reference: ^ACCT("ACCOUNTS",0,{Accounts})
  Row ID Field Name: Accounts      Base on Other Fields? No
                        Based on Fields:
  Generated Routine: mt21
-----

Col      Max.      Multi
No.  Field Name      Description      Data Type  Len.  Uniq.  Req.  Line
-----
1  Accounts      Accounts Row ID      Row ID      16     Yes   Yes   No
2  Account_num    account number      Number      10     No    No    No
3  City_Acct      City of account      Text        24     No    No    No
4  Name_Acct      Name of Account Holder  Name        32     No    No    No
5  Phone_Acct     Phone Number         Number      10     No    No    No
6  Account_Name    Name of Account      Text        24     No    No    No
Press <Return> to continue, <Options> to scroll, <Exit> to Exit

```

Series of Base Tables Report

The Series of Base Tables Report is identical in content to the Base Table (Order by Field Name) Report. The Base Table (Order by Field Name) Report displays information for one base table; the Series of Base Tables Report displays the same information for a series of base tables whose names begin with a similar string. For example, all base tables whose names begin with "Employ" could be displayed using this report.

The Series of Base Tables Report is grouped by base table. It displays basic information about each base table and provides a list of all fields defined for each base table with detailed information about each one.

When you run the Series of Base Tables Report, you see the Runtime Conditions for Report window, as shown below:

```
+-----+
| Run Time Conditions                                     |
| For Report: Attributes for Series of Base Tables       |
|                                                         |
| BaseTabName      Starts With _____              |
| ( From view:  DataDict      with description Base Table Name ) |
|                                                         |
+-----+
```

Run Time Conditions

Press <PF1><PF3> For Help

In the field entitled BaseTabName Starts With, you specify a string that matches the starting characters of the name(s) of one or more base tables. Name strings are case-independent; punctuation characters in name strings are ignored.

To display all base tables defined in the current directory, you may leave this string field blank.

If the string you specify matches none of the base table names, Open M with SQL returns you to the Reports on Data Dictionary menu.

List Table Relationships Report

The List Table Relationships Report lists all base tables defined in the current directory in alphabetical order and lists the designative and characteristic relationships for each base table.

Specifically, it provides the following information about each base table:

- n Base Table Name
- n Description
- n Designated Table(s)
- n Characteristic Table(s)

Below is a sample List Table Relationships Report:

Schema Report For /us/glenm/			
			03 Jan 97
			04:41PM
Base Table	Description	Designated Table(s)	Characteristic Table(s)
Accounts	Accounts base table		Invoices
Copy_Orders	Book orders	Guides	
Docstaff	Information on documentation staff		
Editions	Table of Latest Editions	Guides	
(2)			
Press <Return> to continue, <Options> to scroll, <Exit> to Exit			

List of Views Report

The List of Views Report lists the names and descriptions of all views defined in the current directory. The list is ordered alphabetically by the name of the view.

Below is a sample List of Views Report:

View Name	View Description
Guide_Orders	Documentation Orders
orgpeople	Organizations and Contact People
Press <Return> to Exit	

View -- Sorted by Path Report

The View -- Sorted by Path Report shows the join specification path for each base table included in a selected view and lists all fields included in the view from each base table. It provides basic information about the selected view and detailed information about each of its fields. The report is sorted by join specification path.

The report provides the following basic information about the selected view:

- n View Name
- n View Description
- n Orientation of View (Base Table or Query)
- n Starting Table

For each base table joined in the view, it provides the following information:

- n Base Table Name
- n Join Specification Path from Starting Table

For each field in the selected view, it provides the following information:

- n Field Name in View
- n Field Description in View
- n Field Name in Base Table
- n Field Description in Base Table

When you run this report, you first see the Select a View window, as shown below:

-----Select a View-----

View Name:

View Selection Selecting Press <PF1><PF3> For Help

Enter the name for this View.

Here you must specify the name of the view for which you want to run the report. You may press the <List Choices> key to see a lookup box that lists all views defined in the current directory.

Below is a sample View -- Sorted by Path Report:

===== Definition of View: Guidestaff (For each guide edition, staff who worked) View is Based on Base Table Starting Table: Editions =====			
Joined Base Table: Editions () Join Specification: Editions			
Field Name	Description	Field Name in Base Table	Field Description in Base Table
Main_Author	Main author of edition	Main_Author	Main author of edition
Revision_Date	Date edition printed	Revision_Date	Date edition printed
Version	Product version	Version	Product version

View -- Sorted by Fields Report

The View -- Sorted by Fields Report lists all fields included in a selected view in alphabetical order. It provides basic information about the selected view and detailed information about each of its fields.

The report provides the following basic information about the selected view:

- n View Name
- n View Description
- n Orientation of View (Base Table or Query)
- n Starting Table or View

For each field in the selected view, it provides the following information:

- n Field Name
- n Field Description
- n Base Table Where Located
- n Field Name in Base Table
- n Join Specification Path from Starting Table

When you run this report, you first see the Select a View window, as shown below:

+-----Select a View-----+		
View Name:	<input style="width: 90%;" type="text"/>	
+-----+		
View Selection Selecting Press <PF1><PF3> For Help		
Enter the name for this View.		

Here you must specify the name of the view for which you want to run the report. You may press the <List Choices> key to see a lookup box that lists all views defined in the current directory.

Below is a sample View -- Sorted by Fields Report:

View Definition Information (By Field)				
View Name: Guidestaff				
View Description: Who worked on each edition				
Starting Type: Base Table				
Starting Table/View: Editions				
Printed on 02/18/97 at 12:15				

Field Name	Description	Base Table	Field Name in Base Table	Path
-----	-----	-----	-----	----
Main_Author	Main author of edition	Editions	Main_Author	Editions
Revision_Date	Date edition printed	Editions	Revision_Date	Editions
Title	Name of Guide	Guides	Title	Editions.Guides
Version	Product version	Editions	Version	Editions
~~~~~				
Page: 1				
Press <Return> to Exit				



```

Global Documentation for: ^Guides
=====
^Guides(1,0,{Guides}) -----

Multi Multi-Line
-----
Delimiter Piece Field Name Line? Delimiter
-----
^Guides(1,0,{Guides}) -----
      $c(1)      1      Title      No
                  2      Product     No
                  3      Num_Editions No
                  4      Latest_Edition_Gui No
                  5      Retired      No
                  6      Date_Retired No
                  7      Who_Retired  No

^Guides(1,9,{Latest_Edition_Gui},{Guides}) -----

^Guides(1,"2A",$SALPHAUP({Title}},{Guides}) -----
      $c(1)      1      Title

```





Below is a sample Global Doc -- Map Version Report:

Global Documentation for: ^Guides						
Multi						
Full Reference	Field Name	Node	Piece	Delimiter	Line Field	Maximum Length
^Guides(1,"2A",\$\$ALPHAUP({Title}},{Guides}))	Title		1	\$c(1)		30
^Guides(1,0,{Guides}))	Title		1	\$c(1)		30
	Product		2	\$c(1)	No	6
	Num_Editions		3	\$c(1)	No	16
	Latest_Edition_Gui		4	\$c(1)		16
	Retired		5	\$c(1)	No	4
	Date_Retired		6	\$c(1)	No	12
	Who_Retired		7	\$c(1)	No	25
Press <Return> to continue, <Options> to scroll, <Exit> to Exit						





---

# ***Physical Structure***

---

Chapter 16

## ***Default Physical Structure***

Chapter 17

## ***Relational Definition of an M Database***

Chapter 18

## ***Creating a Customized Map Definition***

---





---

## Default Physical Structure

---

This chapter describes the default physical structure that Open M with SQL generates. Understanding this structure can help prepare you to perform customized mapping. This chapter covers the following topics:

- n Overview of Default Physical Structure page 16-2
- n Examining the Row ID Field Definition page 16-3
- n Accessing the Master Map and Index Maps page 16-10
- n Examining a Default Master Map page 16-12
- n Examining a Default Index Map page 16-22
- n Changing the Global Name in a Default Physical Structure page 16-25
- n Changing a Global's System or Directory page 16-26
- n Updating Default Physical Structure page 16-27

## Overview of Default Physical Structure

When you select default physical structure at the Base Table Definition master window, Open M with SQL automatically creates:

- n A Row ID field.
- n A Master Map, which defines the location of the Row ID and all fields you defined in the base table in the default global ^mdata, unless you selected another global to hold this definition.
- n An Index Map for each field you defined as unique or as a lookup field. Each Index Map defines the location of one or more fields and the Row ID in the same global that was used for the Master Map.

When a row is inserted either via embedded SQL, a query, or a Form Generator form, Open M with SQL uses the global structure defined in the Master Map and Index Maps to generate the appropriate SET commands to update the M global database.

### Examining Default Physical Structure is Optional

When you use default physical structure to define a base table, you do not need to examine the physical structure Open M with SQL generates in order to use that base table in an application.

However, in advance of creating a base table using customized physical structure, it can be useful to examine the global structures Open M with SQL creates when you use default physical structure.

You can examine the default physical structure at any time, but you cannot edit it. All windows are in Inquiry mode.

There are three parts to examining the default physical structure:

- n Examining the definition of the Row ID field.
- n Examining the Master Map.
- n Examining the Index Maps.

To examine the Row ID field, you examine its Field Definition window, and its Row ID Data Type window.

To examine the default global structure, you look at the Map Definition windows for each map Open M with SQL created for the base table. These windows are in Inquiry mode, meaning you cannot edit any values in them.

Field names in this and subsequent chapters are shown enclosed by curly braces, as in {Doc_Name}. This is how you enter field names in customized mapping.



## Converting from Default to Custom Physical Structure

You can initially define a base table using default physical structure and generate the default physical structure. You can then convert the base table to custom physical structure and edit the default physical structure. You can perform this conversion at any time. However, once you convert a base table, you cannot revert from custom physical structure back to default physical structure. See “If You Started Using Default Physical Structure” on page 18-7 in Chapter 18, *Creating a Customized Map Definition*.

## Examining the Row ID Field Definition

In default physical structure, Open M with SQL automatically generates a Row ID field definition. When users enter data into this base table, Open M with SQL automatically provides a unique value for the Row ID field.

Open M with SQL gives the Row ID field the name of the base table. You cannot edit the definition of the Row ID field when using default physical structure.

The Row ID field is always unique. As a result, Open M with SQL puts the value NUMROWS in the Number of Distinct Values field. This is because there will be a different Row ID value for each row in the base table.

**Example** The Row ID field definition for the Guides table from the sample Documentation Tracking application is shown below. Since Open M with SQL names the Row ID field after the table, the field name of the Row ID is {Guides}. Note that the Row ID is unique.

...Field Definition					
Update Features	Conversion/Validation Code	Error/Help Messages	Indexing Options	Copy Field	Additional Options

Base Table Definition	
Base Table Name Guides_____	Description List of all revisions to guides_____

Field Definition	
Field Name Guides_____	Description Guides Row ID_____
Data Type Row ID_____	
Maximum Length 16_	Unique Field? Yes_
Number of Distinct Values NUMROWS_____	
Multi-Line Field? No__	< Multi-Line Options >
Computed Field? No__	

Row ID Definition in Non-Child Tables

For non-child tables, the Row ID is a non-displayable field, which is not based on other fields. It is a Number field, although in Data Type it is defined as Row ID.

**Example** The Row ID data type auxiliary window for {Guides} is shown below. This window is displayed by moving the cursor to the Data Type field, then pressing the <RETURN> key. Because the Guides field is not based on other fields, its value alone constitutes the Row ID for the Guides table:

-----Base Table Definition-----

Base Table Name

Description

Guides

List of all revisions to guides

-----Field Definition-----

Field Name

Description

Guides

Guides Row ID

-----Row ID Data Type-----

Is this Field Displayable? No

Is this Row ID Based on Other Fields? No

Auxiliary Data Type

Number

Fields

(0/)

-----

...Row ID Data Type

Inquiry Mode

Press <PF1><PF3> For Help

Open M with SQL Inserts Value in Row ID Field

When you insert a new row into a non-child table, Open M with SQL generates a default Row ID value in the Master Map. This Row ID value is an integer, generated sequentially starting from 1. Open M with SQL uses an M extrinsic function, \$\$NEXT, to find the subscript of the next node. \$\$NEXT sets the Row ID value counter to 1, if it does not exist. If it does exist, \$\$NEXT increments it by 1, and then returns that value. You can see this code in the M Expression for Default (External) Value field on the Update Features window, as shown below (see “Step 9: Define User Update Features” on page 6-46 in Chapter 6, *Defining Base Table Fields*):

-----Base Table Definition-----+		
Base Table Name	Description	
Guides_____	List of all revisions to guides_____	
-----Field Definition-----+		
Field Name	Description	
Guides_____	Guides Row ID_____	
-----Update Features-----+		
Field Protection Output Only_	Column Number 1	Field ID # 37
Required Field? Yes_____ If it is 'Maybe' Required -		
the M Expression to Determine if it is Required is:		
IF _____		
M Expression for Default (External) Value		
\$\$next( "^mdata( " "GUIDES" " ) " ) _____		
-----+		
...Update Features	Inquiry Mode	Press <PF1><PF3> For Help
Conversion/ Validation Code	Error/Help Messages	

Using a sequentially assigned numeric Row ID, generated by \$\$NEXT, guarantees that there are no blank Row ID values, and that every Row ID value is a unique value.

## Row ID Definition in Child Tables

In the child table created in a characteristic relationship, the Row ID field for each row must logically represent both the child rows and the row in the parent table to which the child rows are joined. To accomplish this, Open M with SQL creates three fields:

- n Row ID field
- n Parent Reference
- n childsub

For information on creating a child table, see “Defining a Child Table” on page 5-11 in Chapter 5, *Defining a Base Table* .

### Row ID Field

The Row ID field in a child table represents a combination of the parent reference and childsub fields. Thus, it does not represent a subscript that physically appears in a global, but rather a logical entity. Its name is the name of the base table, it takes only unique values, and the number of distinct values is equal to NUMROWS.

### Parent Reference Field

The parent reference field is a designative reference to the parent table. Open M with SQL gives this field the name of the parent table. The Description generated for this field consists of the field name, followed by the phrase "Parent Reference", and the column number 0.

### childsub

The childsub field provides a unique identifier for each row associated with a particular row in the parent table. However, it therefore can have the same value for different values of the Parent Reference, so this field is not unique. Open M with SQL names it {childsub}.

### Open M with SQL Assigns Values

When you insert a new row into a child table, Open M with SQL places the Row ID of the corresponding parent table in the parent reference field. It then generates a number in the childsub field. For each new parent reference, it starts the count of {childsub} at 1.

### Example of Row ID Field Definition for Child Tables

In our sample application, we created the base table Editions as a child of Guides. We elected to have Open M with SQL generate a default physical structure for the Editions table. As part of this job, it created the Row ID field, {Editions}, and the two fields on which {Editions} is based. Now that the Editions base table exists, we can look at its definition. The Field Definition selection window for

Editions is shown below. The lookup box contains the three fields Open M with SQL created.

- n The {Guides} field is a designative reference field to the parent table, Guides. Open M with SQL automatically gives it a description which includes the name of the parent table and the phrase "Parent Reference". It has the column number 0, as do all parent designative reference fields in child tables.
- n The Row ID field, {Editions}, has the column number 1, as do all Row ID fields. It is based on {Guides} and {childsub}.
- n The {childsub} field has the column number 2, since Open M with SQL assigns column numbers in sequential order.

-----Base Table Definition-----+		
Base Table Name	Description	
Editions	Editions of Guides	
-----Field Definition-----+		
Field Name	Description	
childsub	Child subscript	2
Editions	Editions Row ID	1
Guides	Guides Parent Reference	0
-----		
...Field Definition                      Selecting                      Press <PF1><PF3> For Help		

First, let's look at the definition for the Row ID field, {Editions}. A child table's Row ID field has the same definition as a simple Row ID field, as shown earlier

in this chapter. However, the information in its Row ID auxiliary window is different. A child table's Row ID field is based on other fields, as shown below:

-----Base Table Definition-----	
Base Table Name	Description
Editions	Editions of Guides
-----Field Definition-----	
Field Name	Description
Editions	Editions Row ID
-----Row ID Data Type-----	
Is this Field Displayable? ____	
Is this Row ID Based on Other Fields? Yes_	Auxiliary Data Type
Fields	Guides
(0/2)	childsub
Guides.Guides    Versions.childsub	
-----	
...Row ID Data Type Inquiry Mode Press <PF1><PF3> For Help	

Now let's look at the fields on which {Editions} is based. The next screen shows that the {Guides} parent reference field is a designative reference to the Guides table.

-----Base Table Definition-----					
Base Table Name	Description				
Editions	Editions of Guides				
-----Field Definition-----					
Field Name	Description				
Guides	Guides Parent Reference				
Data Type Designative Reference ____					
Maximum Length 16_	Unique Field? ____				
-----Designative Reference Data Type-----					
Number	Referenced Table Guides				
Multi					
-----					
Computed Field? No__					
-----					
Update	Conversion/	Error/Help	Indexing	Copy	Additional
...Designative Reference Data TypeInquiry Mode Press <PF1><PF3> For Help					

Finally, let's look at the definition of {childsub}. It is similar to a Row ID field in a non-child table, in that it is a Number type field with a maximum length of 16. It contains an identifier for each child row related to a particular parent reference.

The main difference is it is not unique since it starts with the value 1 for each new parent reference value.

-----Base Table Definition-----					
Base Table Name		Description			
Editionsc_____		Editions of Guides_____			
-----Field Definition-----					
Field Name		Description			
childsub_____		Child subscript_____			
Data Type Number_____					
Maximum Length 16_				Unique Field? ____	
Number of Distinct Values _____					
Multi-Line Field? No__				< Multi-Line Options >	
Computed Field? No__					
...Field Definition		Inquiry Mode		Press <PF1><PF3> For Help	
Update Features	Conversion/ Validation Code	Error/Help Messages	Indexing Options	Copy Field	Additional Options

**Procedure To examine the Row ID field definition:**

1. Press <List Choices> at the Field Definition row selection screen.  
A lookup box containing the defined fields in this base table displays.
2. Use the arrow keys to move the cursor to the Row ID field and press <RETURN>.  
  
The name of the Row ID field is the table name  
  
You see the Field Definition master window.
3. Use the <Tab> key to move the cursor to the Data Type field and press <RETURN>.  
  
You see the Row ID auxiliary window.
4. Press <GETOUT> when you are done examining the Row ID definition to return to the Field Definition row selection window.

# Accessing the Master Map and Index Maps

To examine the physical structure defined in the Master Map or Index Maps, you need to enter the Map Definition form.

**Procedure** To access the Master Map and Index Maps of your base table:

- 1. At the Base Table Definition master window, move the cursor to the <Physical Structure> branching field and press <RETURN>.

You see the Default Physical Structure window, with the cursor at the File name field. The File name field shows the name of the global used in the default global structure. The default value is ^mdata, as shown below. However, this is the one field whose value you may, and should, edit. See “Generating Default Physical Structure” on page 5-15 in Chapter 5, *Defining a Base Table*

-----Base Table Definition-----+

Base Table Name

Docstaff

Description

Information on documentation staff_

-----Default Physical Structure-----+

File name ^mdata

< Fields to Index >

< Examine Structure >

-----+

...Default Physical Structure

Press <PF1><PF3> For Help

Recreate Structure

- 2. Press <TAB> two times to move the cursor to the <Examine Structure> branching field, then press <RETURN>.

If Open M with SQL has not yet generated the default physical structure for this base table, it does so now. See “Generating Default Physical Structure” on page 5-15 in Chapter 5, *Defining a Base Table*

Open M with SQL displays the Map Definition selection window.

- 3. Press <List Choices> to see a lookup box containing the Master Map and all Index Maps.

The highlight bar is on the first map, which is the Master Map. The Master Map always has the name Data Master Map. If there are Index Maps, they





## Examining a Default Master Map

The Master Map defines the global structure that contains the data for this base table.

The path via M to a row of data is called the Full Row Reference. In default physical structure, it is a global reference. This reference is comprised of two subscripts and then the Row ID field (or the fields that comprise the Row ID field in the case of a child table). All other fields in the base table are pieces of the value of that global node, or subscript levels below that node.

### New and Old Master Map Structures

At Version F.10, the global structure of Master Maps and Index Maps was improved to prevent the orphaning of data when importing default structured base tables. Because old-format map structures begin with a number, and new-format map structures begin with a letter (all base table names must begin with a letter), there is no conflict between old and new map names.

- Old (pre-F.10) base tables may continue to be used with their existing map structures. Tables with old and new Master Map formats may coexist and be associated by designative and characteristic relationships. New-format index maps may coexist with an old-format Master Map, and old and new index maps may coexist for the same base table.
- Newly created (F.10 and subsequent) base tables and indexes are automatically assigned the new map structures.
- Existing old-format master maps can be converted to the new format. For conversion procedures, see “Updating Default Physical Structure” on page 16-27.

Assume you are using the default global, `^mdata`, you have defined a base table which is not a child table, and *field2* is a multi-line field.

#### Old Master Map Structure

The old (pre-F.10) global structure for all fields except the multi-line field for the Master Map is:

```
^mdata(irn,0,RowID)=field1^field3^...^fieldn
```

where

*irn*        the **internal reference number** of the base table. Open M with SQL assigns these numbers, starting from 1, in the order in which you create base tables in a particular M namespace.

0        is the constant value of the second subscript

*RowID* is:

- n The value of the Row ID field for parent tables or
- n Comprised of two subscripts
  - the first contains the value of the parent reference
  - the second the value of the childsub field

*fieldn* is the value of a field you defined in the base table.

### New Master Map Structure

The new (F.10 and subsequent) global structure for all fields except the multi-line field for the Master Map is:

```
^mdata (TableName, 0, RowID) = field1 ^ field3 ^ ... ^ fieldn
```

where

*TableName* is the initial user-specified name of the base table, in its internal representation (all uppercase letters, underscores removed). Subsequent changes to the table name are *not* reflected in this master map structure. For example, if you create a base table named Accounts, then change its name to Curr_Accts, this element of the Master Map will remain “ACCOUNTS”. If you subsequently create another base table named Accounts, the Data Dictionary would make this Master Map element unique by appending a number, for example, “ACCOUNTS1”.

0 is the constant value of the second subscript.

*RowID* is:

- n The value of the Row ID field for parent tables or
- n Comprised of two subscripts
  - the first contains the value of the parent reference
  - the second the value of the childsub field

*fieldn* is the value of a field you defined in the base table.

### Multi-line Field Default Structure

A multi-line field is stored in separate nodes below the Full Row Reference, in a storage mode called One Per Node. (One Per Node is the only storage mode available with default physical structure.) For each row, a counter of the number of lines entered in the field in that row is stored one subscript level down, with a subscript value of 0. The values of each line are stored at the same level, at subscript values going from 1 to the number of lines entered in the field.

The global structure of the multi-line field, *field2*, is:

```
^mdata(TableName,0,RowID,1,0)=number_of_lines
^mdata(TableName,0,RowID,1,1)=value_of_line1
^mdata(TableName,0,RowID,1,number_of_lines)=value_of_last_line
```

In the Docstaff table, we changed the name of the global which will store the data from the default value, ^mdata, to the descriptive name, ^Docstaff.

**Old Format:** Docstaff was the sixth table we created in this M database, so it has an irn of 6. The reference to a row is thus ^Docstaff(6,0,{Docstaff}).

**New Format:** Docstaff is the initial name assigned to this base table. The reference to a row is thus ^Docstaff("DOCSTAFF",0,{Docstaff}).

## SET Commands for Master Map Structure

The first SET command assigns values to all fields stored at the first level. For Docstaff, this is all fields except {Street} and {Specialties}, which are multi-line fields. Each value at the first level is a piece of that node, separated by the delimiter "^". (These examples are in New Format. For Old Format, substitute the number 6 (the irn for the base table) for the string "DOCSTAFF".)

```
SET ^Docstaff("DOCSTAFF",0,1)="Morrow,Glenn^6173334444^Cambridge^MA^02142"
```

Open M with SQL then issues SET commands for each multi-line field. It first sets the counter to the number of lines entered. It then issues a SET for the value of each line. For Docstaff, it issues SET commands for the two multi-line fields, {Street} and {Specialties}. Open M with SQL always uses the One Per Node storage structure for multi-line fields when you select default physical structure. These commands look like this:

```
SET ^Docstaff("DOCSTAFF",0,1,1,0)=2
SET ^Docstaff("DOCSTAFF",0,1,1,1)="123 Main Street"
SET ^Docstaff("DOCSTAFF",0,1,1,2)="Apt. 3G"

SET ^Docstaff("DOCSTAFF",0,1,2,0)=2
SET ^Docstaff("DOCSTAFF",0,1,2,1)="FrameMaker"
SET ^Docstaff("DOCSTAFF",0,1,2,2)="Adobe Acrobat"
```

This is an example of SET commands for the second row in Docstaff:

```
SET ^Docstaff("DOCSTAFF",0,2)="Morrow,Glenn^
6173334444^Cambridge^ MA^02142"
SET ^Docstaff("DOCSTAFF",0,2,1,0)=1
SET ^Docstaff("DOCSTAFF",0,2,1,1)="321 Oak Street"
SET ^Docstaff("DOCSTAFF",0,2,2,0)=3
SET ^Docstaff("DOCSTAFF",0,2,2,1)="WordPerfect"
SET ^Docstaff("DOCSTAFF",0,2,2,2)="html"
SET ^Docstaff("DOCSTAFF",0,2,2,3)="visio"
```

For further details on the SET command, see “Step 1: Compose SET Command to Insert Row” on page 18-5 in Chapter 18, *Creating a Customized Map Definition*.

Let's look at how Open M with SQL defines this structure in the Docstaff Master Map.

## Map Definition Master Window

The Map Definition master window for the Docstaff Master Map is shown below. Note that the Global Name is ^Docstaff, not the default ^mdata. Also note that because this base table uses default physical structure the window is in Inquiry Mode, which means you cannot edit field values.

-----Map Definition-----		
Map Name: Data Master Map_____	Master Map (Y/N): Yes_	
Global Name: ^Docstaff_____		
Field(s) Used to Specify the RowID: (0/1) Docstaff.Docstaff		
< Access Path Specifications > < Full Row Reference >		
< Map Data Specifications > < Row ID Specifications >		
Full Row Reference: (0/1)		
{%row} = ^Docstaff("DOCSTAFF",0,{Docstaff})		
-----		
..Map Definition	Inquiry Mode	Press <PF1><PF3> For Help
Advanced Options		

The table below and on the following pages describes the values in the two database fields on the Map Definition master window:

**Table 16-1: Map Definition Master Window Fields**

Field	Description
Map Name	The name of the map you selected, as created by Open M with SQL. In default structure, the master map is always named Data Master Map.
Master Map (Y/N)	The value here is Yes, since we are looking at the Master Map.
Global Name	This is the name of the global in which the base table definition is stored. See "Generating Default Physical Structure" on page 5-15 in Chapter 5, <i>Defining a Base Table</i> . Notice that we changed the global name from the default name, ^mdata, to ^Docstaff.
Field(s) Used to Specify the Row ID	In default physical structure, the Row ID is always one field for non-child tables, and a field based on two other fields for child tables. Open M with SQL creates these fields when it generates the default physical structure of the base table. Open M with SQL gives the Row ID field the name of the base table. The syntax used to describe these fields is table.fieldname. In the non-child Docstaff table, the Row ID is Docstaff.Docstaff. In the child table Editions, the Row ID is the two fields which comprise the Row ID, Guides.Guides and Editions.childsub.
<Access Path Specifications>	Select this branching field to see the Access Levels Open M with SQL defines to reach the Full Row Reference.
<Full Row Reference>	In customized mapping, you select this field to override the default Full Row Reference. In default physical structure, Open M with SQL always uses the default.
<Map Data Specifications>	Select this branching field to see the location of each non-Row ID field in the table.
<Row ID Specifications>	Select this branching field to see how Open M with SQL creates the Row ID field, given a Full Row Reference.

Table 16-1: Map Definition Master Window Fields

Field	Description
Full Row Reference	<p>The Full Row Reference, stored in the variable {%row}, defines the global reference Open M with SQL uses to refer to one complete row in the base table. (In customized structure, it may not be a global reference.)</p> <p>It consists of the global name and two subscripts plus a subscript for each field which comprises the Row ID. In a non-child table, there is one Row ID field. In a child table, there are two fields which comprise the Row ID.</p>
Full Row Reference (continued)	<p>The syntax of the reference is:</p> <p><code>^globalname({L1},{L2},{RowID field1}[,{RowID field2}])</code>.</p> <p>The global name is either the default value ^mdata or the name you enter following the procedure described in “Generating Default Physical Structure” on page 5-15 in Chapter 5, <i>Defining a Base Table</i>.</p> <p>{L1} is either (new format) the initial name you specified for this base table or (old format) the internal reference number (irn) of the table in this M namespace. In the old format, Open M with SQL gives the first table you create in a particular M namespace an irn of 1, the second an irn of 2, and so on. In the above Map Definition screen, the irn of the Docstaff table, as shown in the value of %row, is 6. This means it is the sixth table we created in this M namespace.</p> <p>{L2} is always 0 for a Master Map.</p> <p>For a non-child table, the third level is the name of the Row ID field, enclosed in curly braces. In the above screen, the third subscript for the Docstaff table is {Docstaff}.</p> <p>For a child table, the Row ID field is comprised of two other fields. These fields appear as the {L3} and {L4} subscripts in the Full Row Reference. {L3} is the designative reference to the parent table. This field has the name of the parent table. {L4} points to the row in the child table and is named {childsub}. In our sample application, Editions is a child table of Guides. Its Full Row Reference is:</p> <p><code>%row=({L1},0,{Guides.Guides},{childsub})</code></p>
Advanced Options	<p>In customized mapping, you select this horizontal menu option to add override filing code or define an Index Map as conditional.</p>

Access Path Specifications

In default physical structure, the global name is the starting point for the Access Path. Open M with SQL uses it as the value of the first Access Level's Data Access Expression. The Access Path Specifications master window for the Docstaff Master Map is shown below:

-----Map Definition-----

Map Name: Data Master MapMaster Map (Y/N): Yes_

Global Name: ^Docstaff

Field(s) Used to Specify the RowID: (0/1)  
Docstaff.Docstaff

-----Access Path Specifications-----

Access Level	Data Access Expression	Access Type	Value Expression	Special Access Code
L1	<Expression>		"DOCSTAFF"	<Access Code>
L2	<Expression>		0	<Access Code>
L3	<Expression>		{Docstaff}	<Access Code>
	<Expression>			<Access Code>
	<Expression>			<Access Code>
	<Expression>			<Access Code>

...Access Path SpecificationsInquiry ModePress <PF1><PF3> For Help

The next window shows the Data Access Expression for Access Level 1.

-----Map Definition-----

Map Name: Data Master MapMaster Map (Y/N): Yes_

Global Name: ^Docstaff

Field(s) Used to Specify the RowID: (0/1)  
Docstaff.Docstaff

-----Data Access Expression-----

Implicit Reference at this level:  
{1D} = ^Docstaff

Override Reference:  
You may specify here an override reference to the above:  
{1D} =

...Data Access ExpressionsInquiry ModePress <PF1><PF3> For Help



## Full Row Reference

Open M with SQL creates the Full Row Reference from the Access Path Specifications. The Full Row Reference is expressed in terms of Access Levels ({L1}, {L2}, etc.) at the Full Row Reference window for the Docstaff table, as shown below:

-----Map Definition-----

Map Name: Data Master Map_____Master Map (Y/N): Yes_

Global Name: ^Docstaff_____

Field(s) Used to Specify the RowID: (0/1)

Docstaff.Docstaff

-----Full Row Reference-----

Implicit Full Row Reference: (0/1)

{%row} = ^Doctaff({L1},{L2},{L3})

Override Full Row Reference:

{%row} = _____

...Full Row Reference      Inquiry Mode      Press <PF1><PF3> For Help

The Full Row Reference ({%row} variable) is expressed in terms of fields on the Map Definition master window.

## Map Data Specifications

The Map Data Specifications master window for Docstaff is shown below.

Note that all fields up to the first multi-line field are pieces of the value of the Full Row Reference:

```
^Docstaff("DOCSTAFF",0)=Name^Phone
```

The first multi-line field is in node 1 below the Full Row Reference:

```
^Docstaff("DOCSTAFF",0,1)
```

The remainder of the fields are in node 2 below the Full Row Reference:

```
^Docstaff("DOCSTAFF",0,2)=City^State^Date_of_Hire^Zip^Country
```

Note the Length_of_Service field, which is a Computed field of type Always Equal to the Computation, is not mapped.

-----Map Definition-----

Map Name: Data Master Map_____

Master Map (Y/N): Yes_

Global Name: ^Docstaff_____

Field(s) Used to Specify the RowID: (0/1)

Docstaff.Docstaff

-----Map Data Specifications-----

Field	Node	Piece	Delimiter	Retrieval
Boss_____	_____	1_____	^^_____	< M Code >
Address_____	_____	2_____	^^_____	< M Code >
Phone_____	_____	3_____	^^_____	< M Code >
_____	_____	_____	_____	< M Code >
_____	_____	_____	_____	< M Code >
_____	_____	_____	_____	< M Code >
_____	_____	_____	_____	< M Code >

...Map Data SpecificationsInquiry Mode

Press <PF1><PF3> For Help

Row ID Specifications

The Row ID Specifications window for Docstaff is shown below. The Row ID is one field, and is simply the value of Access Level 3: the field {Docstaff}.

-----Map Definition-----

-----Access Path Specifications-----

Access Level	Access Type	Value Expression
L1	_____	"DOCSTAFF"_____
L2	_____	0_____
L3	_____	{Docstaff}_____
_____	_____	_____
_____	_____	_____

-----Calculate Row ID Given: Access Levels and Fields-----

Field	Expression
Docstaff	{L3}_____
_____	_____
_____	_____
_____	_____

...Row ID Calculation

Inquiry Mode

Press <PF1><PF3> For Help

Child Table Default Master Map

At Version F.10, the default structure master map for a child table can be in either of two forms. If you specify a File Name of ^mdata, or some other global name,

the child table master map structure is the standard default structure, as described above. The following are master map structures for the Patient parent table and the Patient_Visit child table:

```
^mdata( "PATIENT", 0, {Patient})  
^mdata( "PATIENTVISIT", 0, {Patient_Visit})
```

If you specify a file name of ^*parent, then the default master map structure for the child table is a subtree of the parent table's structure. The following are master map structures for the Patient parent table and the Patient_Visit child table:

```
^mdata( "PATIENT", 0, {Patient})  
^mdata( "PATIENT", 0, {Patient.Patient}, "PATIENTVISIT-  
2", 0, {Childsub})
```

The number 2 appended to the child table name indicates the childsub field, which is always defined as column 2 for default structure tables.

## Examining a Default Index Map

When you use default physical structure, Open M with SQL creates an Index Map for:

- n Unique fields
- n Fields used in a lookup
- n Fields you define as index fields

See Chapter 9, *Index Maps*, to learn more about the function of Index Maps, how the Open M with SQL Query Optimizer selects Index Maps, how to index fields, and which fields to index.

### Old and New Global Structures

At Version F.10, the global structure of Master Maps and Index Maps was improved to prevent the orphaning of data when importing default structured base tables. Because old-format map structures begin with a number, and new-format map structures begin with a letter (all base table names must begin with a letter), there is no conflict between old and new map names.

- n Old (pre-F.10) base tables may continue to be used with their existing map structures. New-format index maps may coexist with an old-format Master Map, and old and new index maps may coexist for the same base table.
- n Newly created (F.10 and subsequent) base tables and indexes are automatically assigned the new map structures.
- n Existing old-format master maps can be converted to the new format. For conversion procedures, see “Updating Default Physical Structure” on page 16-27.

### Old Index Map Structure

The old (pre-F.10) global structure for all fields except the multi-line field for an Index Map created under default physical structure is as follows:

```
^mdata( irn, indn, { indexfield1 } [ , { indexfield2 } , { indexfieldn } ] , { RowID } ) = " "
```

where    *irn*            the internal reference number for the base table

*indn*            the value Open M with SQL gives this index table

*indexfield1...indexfieldn* the names of the fields in the Index Map,  
enclosed in curly braces

*RowID* is:

- n The value of the Row ID field for parent tables or
- n Comprised of two subscripts:
  - the first contains the value of the parent reference
  - the second the value of the childsub field

### New Index Map Structure

The new (F.10 and subsequent) global structure for an Index Map is:

```
^mdata(TableName,col,{indexfield1}[,{indexfield2},{indexfieldn}],{RowID})=" "
```

where *TableName* the initial table name of the base table.

*col* Column number of the field(s) in the index map. If multiple fields are specified in the index, this is a series of column numbers separated by hyphens. The Row ID is not included in this series of column numbers.

*indexfield1...indexfieldn* the names of the field(s) in the Index Map, enclosed in curly braces

*RowID* is:

- n The value of the Row ID field for the base table, or
- n Comprised of two subscripts:
  - the first contains the value of the parent reference field
  - the second contains the value of the childsub field

**Example** Assume you are using the default global, ^mdata, you have defined *field1* as a unique field, and you have defined a lookup that is based on 2 fields: *field3* and *field5*. Open M with SQL creates two Index Maps:

Old Format:

```
^mdata(irn,"map1_name",field1,RowID)=" "
^mdata(irn,"map2_name",field3,field5,RowID)=" "
```

New Format:

```
^mdata(TableName,1,field1,RowID)=" "
^mdata(TableName,3-5,field3,field5,RowID)=" "
```

Open M with SQL searches for the row where the value in the index fields matches the values the user enters. It can then take the corresponding Row ID field value and use it to access the row in the Master Map which contains all the data about the entity.

**Example** The Map Definition window of the Index Map for the field Doc_Name in the Docstaff base is shown below. Note that the values in the Global Name and the Fields Used To Specify the Row ID fields are the same as for the Master Map. However, the value in the Master Map? field is No, and, except for the first subscript, which is the base table name, the full row reference is different.

-----Map Definition-----

Map Name: Index Boss 12_____Master Map (Y/N): No__

Global Name: ^Docstaff_____

Field(s) Used to Specify the RowID: (0/1)  
Docstaff.Docstaff

< Access Path Specifications >

< Full Row Reference >

< Map Data Specifications >

< Row ID Specifications >

Full Row Reference: (0/2)  
{%row} = ^Docstaff("DOCSTAFF",3,{Address},{Docstaff})

...Map Definition

Inquiry Mode

Press <PF1><PF3> For Help

Advanced

Options

Index Maps for Name Data Type Fields

Note that Open M with SQL automatically breaks the Doc_Name field into two parts, so you can search either on first name or last name.

## Changing the Global Name in a Default Physical Structure

The default global in which a base table's data will be stored according to default physical structure is ^mdata. You can change the name of this global.

**Caution:** Do not change the global name once you have entered data into this base table.

InterSystems recommends that you change the name of this global to match the name of the base table. It is much easier to work with the data in a table when it is contained in its own separate, easily identified, global. (If portability across systems is an issue, you may wish to limit your global field names to 8 characters.)

### **Procedure** To change the name of the global holding a base table's data:

1. At the Base Table Definition master window, move the cursor to the <Physical Structure> branching field and press <RETURN>.

You see the Default Physical Structure window, with the cursor at the File name field.

2. Press <Erase Field> to remove the default global name, ^mdata.
3. Enter the name of the global you wish to use.

For the Guides table, we use the global name, ^Guides.

4. Press <PROCEED> to return to the Base Table Definition master window.

## Changing a Global's System or Directory

Open M with SQL supports the use of extended global syntax when specifying the file name for the master map of a base table that uses default physical structure. Extended global syntax lets you store a global on a system and/or in a directory other than the current one.

Base tables that share the same global are in the same namespace. For further information on globals and namespaces, refer to the *Open M/SQL System Management Guide*.

To specify a file name for the master map, go to the Base Table Definition window and press <RETURN> on the <Physical Structure> action field. You see the Default Physical Structure window. At the File Name field you specify the global file name for the master map. By default, this is ^mdata (or ^*parent). Here you may specify extended global syntax to store the master map on another system and/or in another directory. The following are examples of extended global syntax:

On ISM systems:

```
^[ "/usr/msql/dir" ]myglobal
```

or

```
^[ "herman", "usr/msql/dir" ]myglobal
```

On DSM or MSM systems:

```
^[ "ISC", "VOL" ]myglobal
```

On DTM systems:

```
^[ "NAMESPACE" ]myglobal
```

These extended global references are shown in bracket syntax; there is also an environment syntax, which uses vertical bar characters. See “Extended Global Reference” on page 18-11 in Chapter 18, *Creating a Customized Map Definition*



## Updating Default Physical Structure

At Version F.10, InterSystems has improved the default structure for Base Tables. Previously, importing default structured Base Tables could have resulted in orphaned data. The new default structure eliminates this problem by using the Base Table name at the time of table creation as the first subscript and the column number as the second subscript.

The general global structure for maps created under default physical structure is as follows:

Old Global Structure for Default Structure Tables (prior to Version F.10)	
Master map	<code>^mdata(irn,0,{RowID})=field1^...^fieldn</code>
Index map	<code>^mdata(irn,indn,{indexfld1}[,{indexfld2}, ...,{indexfldn}],{RowID})=""</code>

New Global Structure for Default Structure Tables (Version F.10 and subsequent)	
Master map	<code>^mdata(TableName,0,{RowID})=field1^...^fieldn</code>
Index map	<code>^mdata(TableName,col,{indexfld1}[,{indexfld2}, ...,{indexfldn}],{RowID})=""</code>

Parameter	Meaning
<i>TableName</i>	Base Table name at the time of table creation
<i>RowID</i>	Value of the RowID for the Base Table
<i>fieldn</i>	Value of the field(s) defined in the Base Table
<i>col</i>	Column number of the field in the index map
<i>indexfldn</i>	Fields to index

### Example

The Default Structure Base Table named “PATIENT” with one index on {DateOfBirth} (column 3) would be mapped with the following Master Map and Index Map:

```
^mdata("PATIENT",0,{Patient})
^mdata("PATIENT",3,{DateOfBirth},{Patient})
```

```

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA OPEN M Developer
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

  UAAAAAAAAAAAAData DictionaryAAAAAAAAAAU
3
3 Base Table Definition 3
3 View Definition 3
3 Compile a Table 3
3 Copy a Base Table Field 3
3 Copy a View 3
3 Change View's Starting Table 3
3 Recreate all Default Structure Maps 3
3 Populate Index Maps for a Table 3
3 Reports on Data Dictionary 3
3
  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAU

                                     Directory: /isc/patient/
Wednesday Sep 04, 1996              Copyright (c) 1993 - InterSystems Corporation

Data Dictionary 09:00AM                                Press <Help> For Help

Press <RETURN> to recreate the structures of all default structure tables.

```

- ```

UAAAAAAAAAAAAAAAAAAAAAAAAAAAABase Table DefinitionAAAAAAAAAAAAAAAAA;
3
3 Base Table Name Description 3
3 RoomNo Room Number 3
3
UAAAAAAAAAAAAAAAAAAAAAAAAAAAADefault Physical
StructureAAAAAAAAAAAAAAAAAAAA;
3
3
3 File name ^mdata_____
3
3
3
3
3 < Fields to Index >
3
3
3 < Examine Structure >
3
3
3
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAUU

```
- ...Default Physical Structure
Press <Help> For Help

Recreate Structure

You are prompted to confirm the map structure recreation. Enter 'OK' to recreate the global structure for your default structure table(s).

**WARNING:**

Recreating the structure of this table can change the map definitions for this table in such a way that any existing data in this table may be lost.  
You must confirm your decision to recreate the map structures of this table by typing 'OK' at the prompt below.  
Type 'OK' to continue, anything else to quit:

**Warning:** If you recreate the structure of a table, you must also modify any existing globals containing data for that table. Once a table's structure has been recreated, it cannot be reversed to the original format.

Compile the base table. See “Compiling a Base Table” on page 5-17 in Chapter 5, *Defining a Base Table* .



---

## Relational Definition of an M Database

---

This chapter describes how to design and define a relational definition of an existing or planned M database. It then provides an overview describing how to define the maps for each base table you define in the relational definition. The topics covered are:

- |                                                  |            |
|--------------------------------------------------|------------|
| n Overview of Creating a Relational Definition   | page 17-2  |
| n Checklist for Creating a Relational Definition | page 17-4  |
| n Steps 1-12 from Checklist                      | page 17-5  |
| n Example Using Enhanced ^SAMPLE Global          | page 17-19 |
| n Example Using ^ACCT Global                     | page 17-22 |
| n Example Using ^FLAVORS Global                  | page 17-29 |

## Overview of Creating a Relational Definition

The Open M with SQL Data Dictionary provides features that allow you to define a relational definition of almost any M global database structure. This capacity allows you to retain your investment in existing data. And although InterSystems recommends that you use default physical structure for new application, you can use customized physical structure if you wish.

### Existing M Database

If you have an existing M database, you can create a relational definition of it by examining its structure. This chapter helps you in that process. Once you design the relational definition — base tables and views — you define them in the Data Dictionary. You must use customized mapping to describe the location of the base tables in your existing M database.

#### You Can Continue Using Existing M Applications

After you define a relational definition of an existing M database in the Data Dictionary, you can continue to use your existing M applications with your database. Then, at your convenience, you add SQL code to your current applications to take advantage of data control you provide within the Data Dictionary base table definitions. You can also create a new user interface to your application consisting of menus, forms and reports you create using the Open M with SQL application development environment.

### New M Database

InterSystems recommends that you use default physical structure when you create a new application. However, the Data Dictionary provides the flexibility to allow you to design your own M global structure, if you wish.

### Examples

As we discuss the steps involved in creating a relational definition of a global, we use a simple global, ^SAMPLE.

At the end of the chapter, we provide three examples of process of creating a relational definition of an M database using the following globals:

- n   Enhanced ^SAMPLE   We change this global so it contains a designative reference.
- n   ^ACCT                   This global contains nested characteristic relationships, a Row ID which is a meaningful data value, constant subscript values, data as pieces of node value, null values in nodes, and indices using efficient lookup technology.
- n   ^FLAVORS               This global illustrates a row which is a piece of node value.

## Checklist for Creating a Relational Definition

The following checklist specifies the steps you perform to create a relational definition of an existing or planned M database using the Data Dictionary. A description of each step appears later in this chapter. For some steps, this chapter gives full details. For the mapping steps, this chapter refers you to Chapter 18, *Creating a Customized Map Definition*.

### Relational Definition Checklist

|   | Step                                                       | Required?   | Action                                                                                                                                                                                         |
|---|------------------------------------------------------------|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| p | 1.                                                         | Yes         | Design the Relational Definition. See “Step 1: Design a Relational Definition” on page 17-5.                                                                                                   |
|   | Then, define each base table in the relational definition: |             |                                                                                                                                                                                                |
| p | 2.                                                         | Yes         | Create the base table. See “Step 2: Create a Base Table” on page 17-9.                                                                                                                         |
| p | 3.                                                         | Optional    | Edit the Row ID field(s). Optional, can accept system-generated default. See “Step 3: Edit Row ID Field(s)” on page 17-10.                                                                     |
| p | 4.                                                         | Yes         | Define one data field. See “Step 4: Define One Data Field” on page 17-13.                                                                                                                      |
| p | 5.                                                         | Yes         | Create preliminary Master Map for fields you’ve defined so far. )Optional if using default physical structure, otherwise required). See “Step 5: Create Preliminary Master Map” on page 17-13. |
| p | 6.                                                         | Recommended | Test Master Map Access Path Specifications with a query. See “Step 6: Test Access Path Specifications” on page 17-14.                                                                          |
| p | 7.                                                         | Yes         | Define the remaining base table fields. See “Step 7: Define Remaining Base Table Fields” on page 17-16.                                                                                        |
| p | 8.                                                         | Yes         | Complete your Master Map definition. Chapter 18, <i>Creating a Customized Map Definition</i> .                                                                                                 |
| p | 9.                                                         | Optional    | Create Index Maps. See Chapter 18, <i>Creating a Customized Map Definition</i> .                                                                                                               |
| p | 10.                                                        | Optional    | Create Conditional Maps. See “Step 10: Create Conditional Maps” on page 17-17.                                                                                                                 |
| p | 11.                                                        | Recommended | Test each map definition with a query.                                                                                                                                                         |
| p | 12.                                                        | Yes         | Complete base table definition. See “Step 12: Complete Base Table Definition” on page 17-19.                                                                                                   |



## Step 1: Design a Relational Definition

An M database consists of globals. A global is a sparse, subscripted array, which holds an application's data. It has three components:

- n Global name
- n Subscripts (optionally)
- n Value (which might be null)

The syntax for a global reference is:

```
^Global_name(subscript1,subscript2...,subscriptn)=value
```

To design a relational definition of a global, you need to:

- n Examine the global

This will allow you to identify fields and potential base tables.

- n Understand the application which creates and uses this global.

This will allow you to determine whether you need multiple tables, whether there is a designative reference or characteristic relationship between tables, and decide on descriptive field names.

**Example** The ^SAMPLE global contains the following data:

```
^SAMPLE( "Jones , Judy" ) = "18Jan51 "  
^SAMPLE( "Rapp , Jeff" ) = "11Dec51 "  
^SAMPLE( "Stone , Jed" ) = "30Aug66 "
```

### Identify Potential Tables

A global will always have at least one base table associated with it. The exception is an index global.

To identify base tables, look for patterns in the global. Patterns can represent repeating data, which you can define as a field in a base table. For instance, you have seen patterns in the ^SAMPLE global, such as subscript values being names. If there is lots of data at a particular global node, you might define a separate table for that node.

The term field refers to a field you define in the relational definition of the global.

### Ignore Indexes

Do not be concerned with globals or global nodes which are indexes at this time. You can recognize index globals because they will contain data that exists elsewhere in the global or in another global. Often, an index node has no value.

Indexes are handled as non-Master Maps in base table definition. These are called Index Maps.

### Subscripts of Similar Type

Look at the values of a particular subscript. If its value changes, but all values are of a similar type, you may have the Row ID, or part of the Row ID, for a base table.

**Example** The ^SAMPLE global has a subscript which contains repeating data, which appear to be people's names.

### Numeric Subscripts and Multi-Line Fields

Numeric subscripts which increment by one from node to node are often used to store data which spans multiple lines. Two types of multi-line storage are directly supported by Open M with SQL, as described in Chapter 6, *Defining Base Table Fields*.

### Values of Similar Type

Look at the values of a node of a given subscript level. Does it contain different values of similar type? If so, this may be a field or fields in a row of data in a base table.

**Example** The value of each node in the ^SAMPLE global is a date.

## Identify Characteristic Relationships

Once you identify fields, you need to clarify whether the fields all belong in one base table, or in several tables that share a characteristic relationship or designative reference. See “Example Using ^ACCT Global” on page 17-22.

### Known vs. Unknown Number of Values

If you have identified a repeating node value which will be a base table field, you need to know if there is an unlimited number of values that can be entered for that field for a given row. Unless it is a type of multi-line structure, you need to create a child table to contain that field. One value will be in each row of the child table, and each row will be related to one row in the parent table.

### Dependent Relationships

See if multiple instances of data are related to nodes with specific subscript values. This is a good indication of parent-child relationships, which are defined in Open M with SQL as a type of implicit join called a characteristic relationship.

## Identify Fields in Each Table

Now that we have resolved the base tables in the ^SAMPLE global, we can pin down the fields in its one base table. We will need to look at the application to help us with this.

We will enclose the names we give fields in our sample base tables in curly brackets, as you do when referencing fields during customized mapping.

**Example** We have determined that the ^SAMPLE global has two data fields, which we will call {Name} and {Date} for now. We do not know what {Date} represents, so we need to examine the application which populates ^SAMPLE. It turns out that it represents date of birth. We can create one base table for this global. We will call it Birthdays, and we will now call the {Date} field {DOB}.

### Identify Designative References

Look for nodes which contain pointers to a global subscripts and/or value. If the pointer is a complete Row ID for one of the tables you have identified, you can define such a structure as a designative reference. See “Example Using Enhanced ^SAMPLE Global” on page 17-19.

## NULL Values in Fields

Once you have identified fields in the base table, you need to understand the meaning of the data in the fields, particularly if some nodes have no data (are NULL).

## Identify the Row ID of Each Table

The Row ID completely identifies one row in a base table. You first need to determine what comprises a row of data: is it a node value, or a node value and all subsequent subscript node levels, or is it a piece of node value? You then determine which field or fields represent the Row ID: that is, uniquely identify the row. Normally, you will base it on subscript values and possibly piece counters.

If the Row ID is based on other fields, the Row ID field itself does not really have a one-to-one match with some part of the global. However, the fields that comprise it do. The fields that comprise the Row ID do not have to be unique; however, their combination must be unique.

Some of the ideas discussed in this section might not be clear immediately. They will become clear once you learn how to map the specific Row ID structure in Chapter 18, *Creating a Customized Map Definition*.

### Full Row is Global Node Value

Generally, the Row ID field or fields are subscripts which identify the global node which is the Full Row Reference. In that case, all other fields are pieces of that node's value, or of a lower level subscript of that node, unless you define computed fields which are always equal to the results of the computation when defining base table fields. See “Two Types of Computed Fields” on page 6-42 in Chapter 6, *Defining Base Table Fields* .

**Example** In the ^SAMPLE global, a row is the value of the node.

### Full Row Contains a Constant Subscript

Sometimes, there will be a subscript in the global which is a constant. This subscript is not part of the Row ID definition, because it does not vary and does not contribute to uniqueness. Rather, it is used when defining field locations during mapping, at the Map Data Specifications window. See Chapter 18, *Creating a Customized Map Definition* .

**Example** The Row ID of the following global is comprised of the subscripts A and C.

```
^SampGlob(A, "B", C)=data1^data2
```

Remember, A and C do not need to be unique. However, their combination must be.

### Full Row is Subscript

Rarely, a table consists of only one field, which is a subscript in the global. This field then doubles as the Row ID.

### Full Row is a Delimited Piece of Node Value

Sometimes, a row is one piece of a node value, separated by delimiters. This is a case where the Row ID must include a field which is not actually in the global. Open M with SQL manages this special field as a piece counter. See “Step 3: Edit Row ID Field(s)” on page 17-10 to learn how to define the piece counter field. See “Step 4: Define Standard Access Path Specifications” on page 18-14 in Chapter 18, *Creating a Customized Map Definition* , to learn how to use this field when describing the final Access Level.

### Full Row is an Undelimited Piece of Node Value

Sometimes, a row is an undelimited part of a node value, in which case you will use the M \$EXTRACT command to describe how to retrieve a row. See “Step 4: Define Standard Access Path Specifications” on page 18-14 in Chapter 18, *Creating a Customized Map Definition* .

## Complex Global Structures

Some global structures are more complex than our sample globals. For instance, data may be kept in more than one global. Open M with SQL can handle such structures in Map Definition. See Chapter 18, *Creating a Customized Map Definition* .

## Step 2: Create a Base Table

Create one of the base tables in your relational definition.

### Define Child Tables

Be certain not to create tables which are child tables directly. Rather, first create them from their parent table by selecting the Implicit Joins horizontal menu option at the Base Table Definition master window. See “Defining a Child Table” on page 5-11 in Chapter 5, *Defining a Base Table* .

### Define Fields Using Default Physical Structure

It is more straight-forward to change to customized mapping right at the start of base table definition, which is the strategy we use in this guide.

However, if you are mapping to a simple global structure, you can define fields while using default physical structure. This causes Open M with SQL to automatically enter all fields in the Master Map. You will need only to delete and re-enter the mapping for each field.

### **Procedure** To create a base table:

1. Name and describe the table.

See “Defining a Base Table” on page 5-4 in Chapter 5, *Defining a Base Table* .

2. Enter *N* at "Use Default Physical Structure?"
3. Define characteristic (parent/child) relationships.

See “Defining a Child Table” on page 5-11 in Chapter 5, *Defining a Base Table* .

### Step 3: Edit Row ID Field(s)

Open M with SQL automatically creates a Row ID field (or fields, in the case of child tables) whether or not you are using default physical structure. Rarely will the Row ID of an existing M database match the default Row ID definition.

You learned how to determine the Row ID for a base table in “Identify the Row ID of Each Table” on page 17-7. You usually need to edit the Row ID field or fields Open M with SQL creates to match the actual Row ID of your base table.

### Change to Customized Structure Before Editing Row ID Field and Mapping

If you started by generating a default physical structure, you need to change the response at Use Default Physical Structure? on the Base Table Definition master window from No to Yes before you can edit the Row ID field or create the base table's customized map definitions.

### Default Row ID Field in Non-Child Tables

Open M with SQL automatically creates a single Row ID field for all non-child tables, whether or not you start your base table creation with default or customized physical structure. Its field name is the base table name, its data type is Row ID, and its maximum length is 16. None of these values are modifiable. The Row ID field also has the following modifiable values, with the defaults shown here:

|                                               |                                         |
|-----------------------------------------------|-----------------------------------------|
| -----Base Table Definition-----               |                                         |
| Base Table Name<br>Accounts_____              | Description<br>accounts base table_____ |
| -----Field Definition-----                    |                                         |
| Field Name<br>Accounts_____                   | Description<br>Accounts Row ID_____     |
| -----Row ID Data Type-----                    |                                         |
| Is this Field Displayable? No__               |                                         |
| Is this Row ID Based<br>on Other Fields? No__ | Auxiliary Data Type<br>Number_____      |
| Fields<br>(0/)                                | _____                                   |
| -----                                         |                                         |
| ...Row ID Data Type                           |                                         |
| Press <PF1><PF3> For Help                     |                                         |

## Default Row ID Fields in Child Tables

You initially create a child table during parent base table definition. At that time, Open M with SQL asks if you want to use default physical structure. Answer *No*.

Open M with SQL automatically creates two fields in the child table: the Row ID field and the parent reference field. The Row ID field of a child table has the following modifiable values, with the defaults shown here:

|                                               |              |                           |
|-----------------------------------------------|--------------|---------------------------|
| -----Base Table Definition-----               |              |                           |
| Base Table Name                               | Description  |                           |
| child                                         |              |                           |
| -----Field Definition-----                    |              |                           |
| Field Name                                    | Description  |                           |
| child                                         | child Row ID |                           |
| -----Row ID Data Type-----                    |              |                           |
| Is this Field Displayable? ____               |              |                           |
| Is this Row ID Based<br>on Other Fields? Yes_ |              | Auxiliary Data Type       |
|                                               |              |                           |
| Fields                                        | parent       |                           |
| (0/2)                                         | childsub     |                           |
| parent.parent    child.childsub               |              |                           |
| -----                                         |              |                           |
| ...Row ID Data Type                           | Inquiry Mode | Press <PF1><PF3> For Help |

You need to add fields to complete the definition of the Row ID and update the definition of the Row ID field.

## Row ID Field is a Single Data Field

Often, the Row ID field in an existing M database contains meaningful data, often of a non-numeric type. In this case, you need to enter the Row ID Data Type window and:

- n Indicate that the Row ID is displayable.
- n Define the Auxiliary Data Type to reflect the type of its data.

**Example** When we create the Birthdays base table, we will edit the definition of the Row ID field, which Open M with SQL names {Birthdays}. We change its name to {Name}. At the Row ID Data Type window, we define it as a displayable field and give the field the auxiliary data type Name.

### Full Row is a Delimited Piece of a Global Node

If a full row is a delimited piece of a global node, you need to define a special field that is one of the fields on which the Row ID is based. Open M with SQL uses this field as a piece counter when accessing rows of data in a loop. You then use this field when you define the final Access Level in your master map. See “Step 4: Define Standard Access Path Specifications” on page 18-14 in Chapter 18, *Creating a Customized Map Definition*.

#### **Procedure** To create a piece counter field as part of Row ID:

1. Add a field named "Piece\_Counter" or some similar descriptive name.
2. Make it Data Type Number, with no decimal places, of format type 1.
3. Enter *N* at Unique Field?
4. Save the field definition.
5. Select the Row ID field.
6. Press <RETURN> at the Data Type field.  
You see the Row ID Data Type window.
7. Enter *Yes* at "Is this Row ID Based on Other Fields?".
8. Add the piece counter field to the list of fields on which the Row ID is based.
9. Save the Row ID field definition.



## Step 4: Define One Data Field

It is wise to test your Row ID field definition and mapping before defining and mapping all fields in the base table. In order to test the mapping of your Row ID fields (which you define in Access Path Specifications), you must define one data field. Follow the steps in Chapter 6, *Defining Base Table Fields*.

## Step 5: Create Preliminary Master Map

Each base table has one and only one Master Map. It defines access to all rows in a base table, and all stored fields in each row. It defines the location of these fields in your M database. Once you have defined the fields in your base table, you can create the Master Map definition.

### You Must Define Row ID

The data in the Master Map is stored by Row ID. To access the data for a particular row in the Master Map, you must identify the Row ID of the row. Conversely, the Row ID is all you need to access that row.

### A Master Map Cannot Be a Conditional Map

You may not define the Master Map to be a conditional map, since conditional maps only contain some rows from a base table, whereas the Master Map controls the insertion of all rows added to the base table.

### How to Create a Preliminary Master Map

To create the preliminary Master Map, you must define:

- n Name of Master Map
- n Global name
- n Access Path Specifications, possibly including:
  - Special Access Code
  - Additional Data Access Variables
- n Full Row Reference
- n Row ID Specifications
- n One data field in Map Data Specifications

To learn how to accomplish the above, see Chapter 18, *Creating a Customized Map Definition*. Be sure to note any issues that pertain specifically to the Master Map.

## Step 6: Test Access Path Specifications

It is always wise to test your Access Path Specifications before defining the location of all fields based on those specifications. You do this by defining and running an SQL query.

**Note:** You must compile your base table before you can use it as a data source for forms or reports.

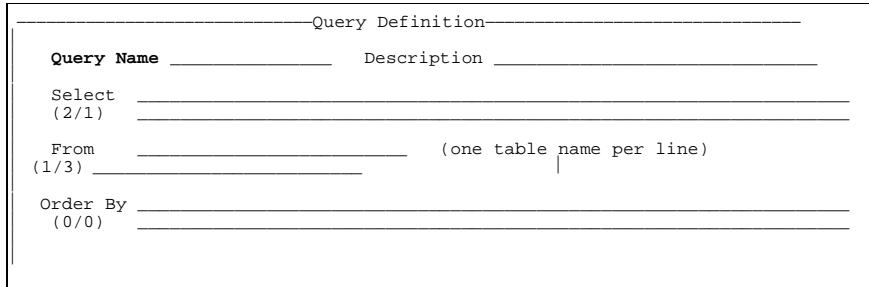
### **Procedure** To define a query to test Access Path Specifications:

1. Go to the Open M with SQL Main Menu.
2. Select the option Queries.
3. Select the option Define Queries.

You see the Query Definition row selection window.

4. Enter a name for your query at the Query Name prompt, such as TEST\_basetablename.
5. Enter a description of the query.

You see the Query Definition master window.



The screenshot shows the 'Query Definition' window. It has a title bar 'Query Definition'. Below the title bar, there are two labels: 'Query Name' and 'Description', each followed by a text input field. Below these, there are three more sections: 'Select' with a counter '(2/1)' and a text input field; 'From' with a counter '(1/3)' and a text input field, followed by the instruction '(one table name per line)'; and 'Order By' with a counter '(0/0)' and a text input field.

6. In the SELECT field, enter the name of the one data field whose location you defined in Map Data Specifications in Step 9.
7. In the FROM field, enter the name of the base table.
8. Move the cursor to the Run this Query? field, enter *Yes* and press <RETURN>.
9. Press <RETURN> to accept the default "No" at the prompt "Compile in the background? No\_" on the message line.

10. In the Device: prompt at the Device Selection window, enter the device to which you want Open M with SQL to send the output, or press <RETURN> to print output to your terminal.
11. Press <RETURN> at the remaining Device Selection fields, then <PROCEED>.

Open M with SQL will send the output to the screen or device you selected. You can verify that the data is being retrieved as you expect.

12. Press <GETOUT> when you are done examining your data.

You return to the Query Definition window.

13. Press <Delete Row> to delete the query.

**Example** The Query Definition to test the Access Path Specifications for the Birthdays base table is shown below. We defined the location of the field {DOB}.

| Query Definition |                                            |
|------------------|--------------------------------------------|
| Query Name       | <i>Birthdays</i>                           |
| Description      |                                            |
| Select           | <i>DOB</i>                                 |
| (0/1)            |                                            |
| From             | <i>Birthdays</i> (one table name per line) |
| (0/1)            |                                            |
| Where            |                                            |
| (0/)             |                                            |
| Order By         |                                            |

The next window shows the output from this query:

| Birthdays Row ID | DOB     |
|------------------|---------|
| Jones, Judy      | 18Jan51 |
| Rapp, Jeff       | 11Dec51 |
| Stone, Jed       | 30Aug66 |

## Step 7: Define Remaining Base Table Fields

Once you have determined that your preliminary Master Map definition is successful, you return to the Base Table Definition form and define the remaining fields. See Chapter 6, *Defining Base Table Fields*, for details on this process.

### Some Fields You Define May Not Be in M Database

The Master Map defines where in your existing M database the data in the fields is located. The Master Map should define a location for all fields in your table except:

- n Computed fields which are always equal to the results of the computation (see “Two Types of Computed Fields” on page 6-42 in Chapter 6, *Defining Base Table Fields* ).
- n Piece counter fields you create when a full row is a piece of a node value

## Step 8: Complete Master Map Definition

Complete your Master Map definition by entering the names of the remaining data fields at the Map Data Specifications window and defining their location in your M database. See “Step 9: Define Map Data Specifications” on page 18-45 in Chapter 18, *Creating a Customized Map Definition* .

## Step 9: Create Index Maps

Perhaps your M database contains global structures which are indexes to speed access to portions of your data. If so, you can define these global structures in Index Maps when mapping the base table which contains the fields in the index.

To learn how to define an Index Map, see Chapter 18, *Creating a Customized Map Definition* . Be sure to read the notes that pertain specifically to the creation of Index Maps.

### Row ID Must Exist in Global Index Structure

When you define an index map, the Row ID of the Master Map must be identified by the subscripts and field data stored in the Index Map, as specified in the Row ID Specifications window. (See “Step 8: Edit Row ID Specifications” on page 18-43 in Chapter 18, *Creating a Customized Map Definition* .)

If all fields required to form the Row ID are not present in the index global structure and they cannot be computed from index information, then the structure does not qualify as an Index Map for the table.

## Step 10: Create Conditional Maps

Perhaps your existing M database contains a structure which contains only a subset of rows and/or fields in one of the base tables you define. In other words, it holds only those rows, or fields of those rows, which meet a certain condition. These maps are called Conditional Maps. Conditional Maps are a special form of Index Map.

To learn how to define an Index Map as a Conditional Map, see “Step 11: Define an Index Map as a Conditional Map” on page 18-58 in Chapter 18, *Creating a Customized Map Definition* .

## Step 11: Test Your Map Definitions

When you complete a map definition, it is wise to test it again prior to compiling the base table. You may want to try a more rigorous query which includes a condition.

**Example** This query selects only some fields and includes a condition:

```
SELECT field_a,field_k
FROM table
WHERE field_c=<value>
```

**Procedure** To define a query to test a customized map definition:

1. Go to the Open M with SQL Main Menu.
2. Select the option Queries.
3. Select the option Define Queries.

You see the Query Definition row selection window.

4. Enter a name for your query at the Query Name prompt, such as TEST\_basetablename.
5. Enter a description of the query.

You see the Query Definition master window.

6. To test the Master Map, enter "\*" (an asterisk without the quotes) in the SELECT field.

This will cause Open M with SQL to display all fields in the table. As a result, it will use the Master Map definition, since that is the map which handles all fields in the base table.

If there are many fields, you may have one of the following problems:

- The column headings and data will wrap, which could make the output difficult to read.
- The query won't compile.

In this case, it may make sense to define several queries, and actually name the 4 or 5 fields you want displayed in each.

7. To test an Index Map, select all the fields in that Index Map and no other fields.

This does not guarantee that the Open M with SQL Query Optimizer will use the index map to locate the fields; however there is a good chance that it will.

8. In the FROM field, enter the name of the base table.
9. In the WHERE field, add a condition.
10. Move the cursor to the Run this Query? field, enter *Yes* and press <RETURN>.
11. Press <RETURN> to accept the default "No" at the prompt "Compile in the background? No\_" on the message line.
12. In the Device: prompt at the Device Selection window, enter the device to which you want Open M with SQL to send the output, or press <RETURN> to print output to your terminal.
13. Press <RETURN> at the remaining Device Selection fields, then <PROCEED>.

Open M with SQL will send the output to the screen or device you selected. You can verify that the data is being retrieved as you expect.

14. Press <GETOUT> when you are done examining your data.

You return to the Query Definition window.

15. Press <Delete Row> to delete the query.

## Step 12: Complete Base Table Definition

Now you can complete your base table definition: lookups, approximate number of rows, base table validation, base table triggers, routine name prefix. You then compile your table. These are Steps 5, 8, 9, 10, 12 and 13 from the Base Table Definition Checklist in Chapter 5, *Defining a Base Table*.

If you have done your mapping in such a way that Open M with SQL does not generate automatic filing code for some fields in the table, you now enter override filing code on a field level in Base Table Triggers for those fields.

## Example Using Enhanced ^SAMPLE Global

We will now enhance the ^SAMPLE global in order to illustrate how to identify, define and map designative references. Let's say we want to keep track of the astrological sign of each of our friends. Since this information would be very repetitive -- many people will share the same sign of the zodiac -- it makes sense to store the signs in a separate ^ZODIAC global and then add a pointer field in the ^SAMPLE global.

### Description of Enhancement

We create a global called ^ZODIAC. The complete data it holds looks like this:

```
^ZODIAC(1)="Aries"
^ZODIAC(2)="Taurus"
^ZODIAC(3)="Gemini"
^ZODIAC(4)="Cancer"
^ZODIAC(5)="Leo"
^ZODIAC(6)="Virgo"
^ZODIAC(7)="Libra"
^ZODIAC(8)="Scorpio"
^ZODIAC(9)="Sagittarius"
^ZODIAC(10)="Capricorn"
^ZODIAC(11)="Aquarius"
^ZODIAC(12)="Pisces"
```

We add a piece to the value of each ^SAMPLE global node. This piece contains the subscript of the node in ^ZODIAC which contains the sign of the person whose name is the subscript of the node in ^SAMPLE. For example:

```
^SAMPLE("Rapp,Jeff")=30Aug66^6
```

## Identify Potential Tables

### Subscripts and Values of Similar Type

The information in the ^ZODIAC global clearly lends itself to being defined a separate table, which we can call *Zodiac*.

## Identify Fields in Each Table

The Zodiac base table contains two fields, described in the table below.

**Table 17-1: Zodiac Table**

| Field Name | Description                                        |
|------------|----------------------------------------------------|
| {Zodiac}   | Subscript value, which is an integer from 1 to 12. |
| {Sign}     | A sign of the zodiac.                              |

The Birthdays table now contains one additional field.

**Table 17-2: Birthdays Table**

| Field Name | Description                                                                   |
|------------|-------------------------------------------------------------------------------|
| {Sign_DOB} | A pointer to the sign of the zodiac associated with a person's date of birth. |

### Identify Designative References

The new field {Sign\_DOB} in the table Birthdays is a designative reference to the table *Zodiac*, because it contains the full Row ID of that table.

## Identify the Row ID of Each Table

We already know the Row ID of the Birthdays table is a person's name, which is the value of its one subscript.

### Full Row is Global Node Value

In the *Zodiac* table, the data is a node value in the ^ZODIAC global. Therefore, the Row ID is the value of the global's one subscript, which is an integer from 1 to 12.

## Create a Base Table

We need to create the new base table, *Zodiac*.



## **Edit Row ID Field**

{Zodiac}, the Row ID of the ZODIAC table, matches the default Row ID created by Open M with SQL, so we don't need to edit its field definition.

## **Define One Data Field**

We define the field {Zodiac} as a text field.

We edit the definition of the base table Birthdays by adding the field definition for {Sign\_DOB}. We define its data type as designative reference. The designated table is Zodiac, which we must have already created.

## **Create Master Map**

See Chapter 18, *Creating a Customized Map Definition*, for information on how to create a Master Map.

### **Zodiac Table**

Use {Zodiac} to define the Access Path Specifications for the Zodiac table. Enter Sub as the Access Type and enter {Zodiac} as the Value Expression for Access Level 1.

Define {Sign} in Map Data Specifications by simply entering the name of the field.

### **Edit Map for Birthdays**

Edit Map Data Specifications. {DOB} is now Piece 1, with a delimiter of "^". {Sign\_DOB} is Piece 2, with a delimiter "^".

## Example Using ^ACCT Global

We use the ^ACCT global to illustrate a number of structures the Data Dictionary handles, including the following:

- n Data as pieces of pieces of node value (Invoice\_Date and Payment\_Date)
- n Constant subscripts ("Phone")
- n Limitations on Access Path Specifications
- n Indexes (The "CITY" and "n" nodes)
- n Storing index data in ALPHAUP format
- n Nested child tables

### Description of ^ACCT Global

#### Syntax

The ^ACCT global contains three subscript levels and six types of nodes. These nodes are:

```
^ACCT( account_number ) = City ^ Account_Name  
^ACCT( account_number , "Phone" ) = Phone_Number  
^ACCT( account_number , invoice_number ) = Invoice_date : Payment  
_ Date ^ Total  
^ACCT( account_number , invoice_number , item_number ) = Quantity  
^ Item_Name ^ Cost  
^ACCT( "CITY" , City , account_number ) = " "  
^ACCT( "n" , Account_Name ) = account_number ^ Account_Name
```

## Sample Data

Below is some sample data in the ^ACCT global:

```

^ACCT(100)=Butte^Communications Consultants
^ACCT(100,"Phone")=222-3333
^ACCT(100,1000)=55740:55759^1000.00
^ACCT(100,1000,1)=100^Paper Clip^100.00
                2)=100000^White Lined Paper^900.00
^ACCT(100,2000)=55496:^1.00
^ACCT(100,2000,1)=1^Paper Clip^1.00
^ACCT(200)=Los Angeles^Clean Air Lovers
^ACCT(200,1)=55708:55730^10.00
^ACCT(200,1,1)=3^Smog (cubic yds)^10.00
^ACCT("CITY","Butte",100)=" "
^ACCT("CITY","Los Angeles",200)" "
^ACCT("n",CLEANAIRLOVERS)=200^Clean Air Lovers
^ACCT("n",COMMUNICATIONSCONSULTANTS)=100^Communications
Consultants

```

Note that the date fields (Invoice\_Date and Payment\_Date) are stored in the same format as the 5-digit \$HOROLOG date value. \$HOROLOG is an M function which stores a number that represents the current date and time. The first 5 digits represent the date.

Note also that the node with "n" as the first subscript stores Account\_Name twice, in two different formats. In the subscript, it is in uppercase and stripped of punctuation and spaces. This is equivalent to the effect of Open M with SQL \$\$ALPHAUP function. The format in the node value is the actual name as entered by a user. When we discuss defining Access Path Specifications for Index Maps in Chapter 18, *Creating a Customized Map Definition*, you will see how this is useful.

## Identify Potential Tables

### Ignore Indexes

In the ^ACCT global, you can see that two nodes are indexes because they repeat account name, city and number. The first index has a null node value; the second index has the Row ID as the node value. Below are 2 sample nodes which illustrate the indexes in the ^ACCT global for the account "Communication Consultants".

```

^ACCT("CITY","Butte",100)=" "
^ACCT("n","COMMUNICATIONSCONSULTANTS")=100^Communications
Consultants

```

### Locate Subscripts of Similar Type

The ^ACCT global has nodes with 1, 2 and 3 subscripts. The first subscript of the ^ACCT global has multiple values, some of which are of similar type: integers. (The "CITY" and "n" values are clearly different. We identified them earlier as indices.)

These first subscript integers represent account numbers. We will define these nodes as a base table named "Accounts." The nodes with two subscripts, where the first subscript continues to be constant and the second subscript is a number which varies, represent invoices of a particular account. The second subscript itself is an invoice number. We will make a base table named "Invoices." The nodes with three subscripts are line items for the invoice identified by the first and second subscripts. The third subscript is a line item number. We will make a table named "Line Items."

### Values of Similar Type

In the ^ACCT global, the value of the first level node is two pieces:

```
^ACCT(100)=Butte^Communications Consultants
```

The first piece is a city, the second a name. These would be fields describing an account, which is identified by the value of the first level subscript.

Information for the "Accounts" table is found in two subscript levels of the ^ACCT global. An example of these nodes follows:

```
^ACCT(100)=Butte^Communications Consultants
^ACCT(100,"Phone")=222-3333
```

### Identify Characteristic Relationships

In the ^ACCT global, the line item nodes make no sense without the matching invoice node. Therefore, the "Line Items" table is in an existence-dependent relationship to the "Invoices" table. This meets the criteria for a parent-child relationship, with "Invoices" being the parent table and "Line Items" being the child table. Likewise, an invoice only makes sense if it is associated with an account. Thus, "Invoices" is a child table of the "Accounts" table.

### Identify Fields in Each Table

#### Accounts Table

Accounts contains four fields, described in the table below. The entry in the "Value" column is from these data nodes:

```
^ACCT(100)=Butte^Communications Consultants
^ACCT(100,"Phone")=222-3333
```

**Table 17-3: Accounts Table**

| Field Name    | Value                      | Description                                                                |
|---------------|----------------------------|----------------------------------------------------------------------------|
| {Account_num} | 100                        | Account number, which is the first-level subscript.                        |
| {City_Acct}   | Butte                      | City where account is located, which is the first piece of the node value. |
| {Name_Acct}   | Communications Consultants | Name of account, which is the second piece                                 |
| {Phone_Acct}  | 222-3333                   | Phone number, which is the value of the node with second subscript "Phone" |

**Invoices Table**

Information in the Invoices table is found in the ^ACCT node with two subscripts, as in this example node:

```
^ACCT(100,1000)=55740:55759^1000.00
```

It contains 5 fields, described in the table below.

**Table 17-4: Invoice Table**

| Field Name     | Value   | Description                                                                               |
|----------------|---------|-------------------------------------------------------------------------------------------|
| {Account_num}  | 100     | Account number, which is the first-level subscript                                        |
| {Invoice_num}  | 1000    | Invoice number, which is the second-level subscript                                       |
| {Invoice_Date} | 55740   | Date of invoice, which is the first piece within the first piece of the node value        |
| {Payment_Date} | 55759   | Date payment received, which is the second piece within the first piece of the node value |
| {Total}        | 1000.00 | Total amount of invoice, which is the second piece of the node value                      |

**Line Items Table**

The "Line Items" table is based on the ^ACCT node with three subscripts, as in this example node:

```
^ACCT(100,1000,1)=100^Paper Clip^100.00
```

It contains 6 fields, described in the table below.

**Table 17-5: Line Items Table**

| Field Name     | Value      | Description                                                                   |
|----------------|------------|-------------------------------------------------------------------------------|
| {Account_num}  | 100        | Account number, which is the first-level subscript                            |
| {Invoice_num}  | 1000       | Invoice number, which is the second-level subscript                           |
| {Lineitem_num} | 1          | Line Item number, which is the third-level subscript                          |
| {Qty_Ordered}  | 100        | Quantity of item ordered, which is the first piece of the node value          |
| {Name_Item}    | Paper Clip | Name of item ordered, which is the second piece of the node value             |
| {Totalcost}    | 100.00     | Total cost of quantity of ordered, which is the third piece of the node value |

## NULL Values in Fields

One node in the ^ACCT global contains a piece with a null value (between the colon (: ) and the carat (^):

```
^ACCT(100,2000)=55496:^1.00
```

There are similar nodes which contain a date in that field:

```
^ACCT(200,1)=55708:55730^10.00
```

Research into the application indicates that the null field represent the payment receipt date. The value is null until payment is received. It will be important not to make this a required field in the Data Dictionary definition, since the value of the field is unknown at initial input and there is significance to a NULL value.

## Identify Row ID of Table

### Full Row is Value of Full Row Reference Node

We defined an Accounts table to describe some of the data in the ^ACCT global. Each row in that table is identified by the field {Account\_num}, which is the value of the first subscript. This is the Full Row Reference for the Accounts table. Two fields in the row, {City\_Acct} and {Name\_Acct}, are located at that node.

```
^ACCT(100)=Butte^Communications Consultants
```

Full Row Contains a Constant Subscript

The third field in the Accounts table, {Phone\_Acct}, is the value of the node one subscript level down, where the second subscript is the constant, "Phone":

```
^ACCT(100,"Phone")=222-3333
```

The "Phone" subscript is not part of the Row ID. However, it is used to identify the location of the field {Phone\_Acct}, which is 222-3333 in the above node.

Edit Row ID Fields

We created the Invoices base table as a child table of the Account base table and entered *No* at the Use Default Structure field. Therefore, Open M with SQL created a Row ID field and a Parent Reference field based on the Row ID of the Account table. However, it did not create a childsub field.

In the ^ACCT global, the identifier of a child row is an invoice number. Therefore, we create a field to identify each child row. Instead of giving it the name childsub, we gave it a informative name and description, as shown below. We gave it the data type Number, described as an integer, since an invoice number in this application is an integer number field.

-----Base Table Definition-----

|                 |                            |
|-----------------|----------------------------|
| Base Table Name | Description                |
| Invoices_____   | Invoices for Accounts_____ |

-----Field Definition-----

|                                 |                        |
|---------------------------------|------------------------|
| Field Name                      | Description            |
| Invoice_num_____                | Invoice number_____    |
| Data Type Number_____           |                        |
| Maximum Length 16_              | Unique Field? No__     |
| Number of Distinct Values _____ |                        |
| Multi-Line Field? No__          | < Multi-Line Options > |
| Computed Field? No__            |                        |

...Field Definition      Unsaved Data      Press <PF1><PF3> For Help

|                 |                             |                     |                  |            |                    |
|-----------------|-----------------------------|---------------------|------------------|------------|--------------------|
| Update Features | Conversion/ Validation Code | Error/Help Messages | Indexing Options | Copy Field | Additional Options |
|-----------------|-----------------------------|---------------------|------------------|------------|--------------------|

Open M with SQL created a standard Row ID field. It already is based on the parent reference. We must edit it to base it also on {Invoice\_Num}.

|                                               |                              |
|-----------------------------------------------|------------------------------|
| -----Base Table Definition-----               |                              |
| Base Table Name                               | Description                  |
| Invoices_____                                 | Invoices for Accounts_____   |
| -----Field Definition-----                    |                              |
| Field Name                                    | Description                  |
| Invoices_____                                 | Invoices Row ID_____         |
| -----Row ID Data Type-----                    |                              |
| Is this Field Displayable? No_____            |                              |
| Is this Row ID Based<br>on Other Fields? Yes_ | Auxiliary Data Type<br>_____ |
| Fields                                        | Accounts_____                |
| (0/2)                                         | Invoice_num_____             |
| Accounts.Accounts    Invoices.Invoice_num     |                              |
| -----                                         |                              |
| ...Row ID Data Type                           | Unsaved Data                 |
| Press <PF1><PF3> For Help                     |                              |



## Example Using ^FLAVORS Global

The ^FLAVORS global is useful to illustrate the following Data Dictionary features to handle its structure, such as:

- n Characteristic relationship (parent-child tables)
- n Row ID is displayable and of type text
- n Each row is a "^" piece of a node value

### Description of ^FLAVORS Global

#### Syntax

The ^FLAVORS global has one subscript level. We can summarize the ^FLAVORS global as follows:

```
^FLAVORS(first_name)=flavor1^flavor2^flavor3
```

#### Sample Data

The first four nodes of the ^FLAVORS global look like this:

```
^FLAVORS("Arthur,Art")=Vanilla^Chocolate^Coffee
^FLAVORS("Boyle,Bob")=Chocolate^Coffee^Strawberry
^FLAVORS("Coyle,Cari")=Vanilla^Mint^Pistachio
^FLAVORS("Derby,Diane")=Coffee^Peach^Mocha
```

The ^FLAVORS global is inefficient in its current form: it contains much data redundancy and thus wastes space on disk. The values Vanilla, Chocolate and Coffee are all stored multiple times. A better design might be to have two globals, one which contains all flavors, and the other which contains a person's name and pointers to their favorite flavors in the other global.

### Identify Fields and Tables

#### Subscripts of Similar Type

The ^FLAVORS global subscript has multiple values of similar type -- first name. It can identify a row in a base table, which we will call "People."

#### Values of Similar Type

In the ^FLAVORS global, the value of each node is three flavors:

```
^FLAVORS("Art")=Vanilla^Chocolate^Coffee
```

There are two ways to define this node value relationally:

- n You can make a base table named "FavFlavors" which is a child table of (dependent on) the People table, since each set of flavors is associated with a particular name in the People table.
- n Since there always seem to be three flavors, you could include three flavor fields in the "People" base table.

The next section helps you decide whether to define the ^FLAVORS globals with one or two base tables.

### Identify Characteristic Relationships

#### Known vs. Unknown Number of Values

When we last visited the ^FLAVORS global, we were unsure whether or not to create two base tables. We need to know whether:

- n Users enter a name, followed by up to, but no more than, three favorite flavors, or
- n Users enter a name, followed by as many flavors as they wish

The sample data we examined showed only three flavors per person. But to be sure, we must examine the data entry code in the existing or planned application.

If you learn users enter a maximum of three flavors, you add 3 flavor fields to the People table. You might call them Choice\_1, Choice\_2 and Choice\_3.

However, if users can enter more than three flavors for a particular person, you need to create two tables. The first, People, contains only the name field. We will also create a second, child table, named FavFlavors. Each row of FavFlavors will contain one flavor enjoyed by the person in the corresponding row of the People table.

We now have defined the two tables in our ^FLAVORS example: People and FavFlavors. They share a characteristic relationship, with FavFlavors being a child table of People.

## Identify Fields in Each Table

The People table has just one field:

**Table 17-6: People Table**

| Field Name | Description                                                                                                                                                                                                         |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {People}   | Name. We will define this field as data type Row ID. Since it contains data in the format used by the Name data type, we give it the auxiliary data type Name. (See Chapter 6, <i>Defining Base Table Fields</i> .) |

The FavFlavors table, which is also mapped to the ^FLAVORS global, has two fields we have identified so far. These fields are described in the table below.

**Table 17-7: FavFlavors Table**

| Field Name | Description                                                  |
|------------|--------------------------------------------------------------|
| {Flavor}   | One of the favorite flavors of a person in the People table. |
| {People}   | Identifies the person associated with the flavor.            |

In the next section, which discusses how to identify the Row ID of each table, we will see we need to create a third field in the FavFlavors table.

## Identify Row ID of Table

### Full Row is Subscript

The People table has only one field, {People}. Since there is only one field, it must be the Row ID field.

### Full Row is Delimited Piece of Node Value

The child table, FavFlavors, must have the same field as the People table as its Row ID: namely, the first name subscript. However, since each flavor is a delimited piece of the node value, you need to define a special field which is a piece counter.

Edit the Row ID Field(s)

Row ID Field is a Data Field

In the People base table, the Row ID is a single, meaningful field, which is the person's first name. As such, we define it as displayable, and indicate that its auxiliary data type is Text, as shown below:

-----Base Table Definition-----+

Base Table Name

Description

People

People in ^FLAVORS global

-----Field Definition-----+

Field Name

Description

People

People Row ID

-----Row ID Data Type-----+

Is this Field Displayable? Yes\_

Is this Row ID Based on Other Fields? No\_\_

Auxiliary Data Type Name

Fields (0/)

-----+

...Row ID Data Type

Unsaved Data

Press <PF1><PF3> For Help

Full Row is a Delimited Piece of a Global Node

In the FavFlavors child table, each row is a piece of a global node. Therefore, we needed to change the default definition of the childsub field to reflect its role in the Row ID. We named it {Piece\_Counter} and changed its description.

| -----Base Table Definition----- |                            |                                |                  |                           |                    |
|---------------------------------|----------------------------|--------------------------------|------------------|---------------------------|--------------------|
| Base Table Name                 |                            | Description                    |                  |                           |                    |
| FavFlavors_____                 |                            | People's Favorite Flavors_____ |                  |                           |                    |
| -----Field Definition-----      |                            |                                |                  |                           |                    |
| Field Name                      |                            | Description                    |                  |                           |                    |
| Piece_Counter_____              |                            | Piece Counter for Row ID_____  |                  |                           |                    |
| Data Type Number_____           |                            |                                |                  |                           |                    |
| Maximum Length 16_              |                            | Unique Field? No__             |                  |                           |                    |
| Number of Distinct Values _____ |                            |                                |                  |                           |                    |
| Multi-Line Field? No__          |                            | < Multi-Line Options >         |                  |                           |                    |
| Computed Field? No__            |                            |                                |                  |                           |                    |
| ...Field Definition             |                            | Unsaved Data                   |                  | Press <PF1><PF3> For Help |                    |
| Update Features                 | Conversion/Validation Code | Error/Help Messages            | Indexing Options | Copy Field                | Additional Options |

We redefined its Row ID field, FavFlavors, to be based on the renamed childsub field, {Piece\_Counter}, as well as the parent reference field, People.

| -----Base Table Definition-----            |                    |                                |  |                           |  |
|--------------------------------------------|--------------------|--------------------------------|--|---------------------------|--|
| Base Table Name                            |                    | Description                    |  |                           |  |
| FavFlavors_____                            |                    | People's Favorite Flavors_____ |  |                           |  |
| -----Field Definition-----                 |                    |                                |  |                           |  |
| Field Name                                 |                    | Description                    |  |                           |  |
| FavFlavors_____                            |                    | FavFlavors Row ID_____         |  |                           |  |
| -----Row ID Data Type-----                 |                    |                                |  |                           |  |
| Is this Field Displayable? No__            |                    |                                |  |                           |  |
| Is this Row ID Based on Other Fields? Yes_ |                    | Auxiliary Data Type _____      |  |                           |  |
| Fields (3/2)                               | Piece_Counter_____ |                                |  |                           |  |
| People.People    FavFlavors.Piece_Counter  |                    |                                |  |                           |  |
| ...Row ID Data Type                        |                    | Unsaved Data                   |  | Press <PF1><PF3> For Help |  |



---

## Creating a Customized Map Definition

---

This chapter describes how to create customized maps for base tables (Steps 5, 8, 9 and 10 from the "Checklist for Creating a Relational Definition" in Chapter 17, *Relational Definition of an M Database*). These maps point to globals in your existing M database or new M database design. The topics covered are:

- n Overview of Customized Map Definition page 18-2
- n Map Definition Checklist page 18-3
- n Field Names for Map Definition page 18-4
- n Checklist Steps 1-11 Described page 18-5

## Overview of Customized Map Definition

If you define a base table with custom (non-default) physical structure, you need to perform customized mapping for that base table. You do this at the Map Definition form, accessed from the Base Table Definition master window. For map definition with default physical structure, see Chapter 16, *Default Physical Structure*.

Once you have created all base table fields and lookups, you are ready to map the base table to an existing or designed M global, global node, or part of a node. First, you create the Master Map. You then create Index Maps. You may wish to create some index maps as conditional Index Maps.

This chapter contains a “Map Definition Checklist” on page 18-3 and detailed steps that make it as easy as possible to define customized maps for your base tables.

1. Make copies of the Map Definition Checklist. Use this list to check off each step as you define each map in your base table.
2. Each step in the checklist is described in this chapter. Read the conceptual information and then follow the procedure to complete the step.

## Sample Globals and Base Tables

We introduced the globals, ^SAMPLE, ^FLAVORS and ^ACCT, and the base tables that provide a relational view to them, in Chapter 17, *Relational Definition of an M Database*. We continue to use these examples in this chapter.



## Map Definition Checklist

The Map Definition Checklist specifies all the required and optional user steps in the map definition procedure. Optional steps are designated "Opt" in the Required? column; highly recommended optional steps are designated "Rec". There are some differences between creating a Master Map and an Index Map for a base table. The steps outlined in this chapter point out those differences.

### Map Definition Checklist

|   | Step | Required? | Action                                                                                                                                    |
|---|------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------|
| p | 1    | Rec       | Compose a SET command that puts data into a row in the base table. See page 18-5.                                                         |
| p | 2    | Yes       | Enter the Map Definition form. See page 18-6.                                                                                             |
| p | 3    | Yes       | Enter the name of the global in which data for this base table is stored. See page 18-11.                                                 |
| p | 4    | Yes       | Define how to reach a row of data in the M global in a standard manner on the Access Path Specifications form. See page 18-14.            |
| p | 5    | Opt       | If necessary, provide Special Access Code on the Access Path Specifications form. See page 18-29.                                         |
| p | 6    | Opt       | If necessary, define Additional Data Access Variables. See page 18-39.                                                                    |
| p | 7    | Opt       | If necessary, provide an override to the Full Row Reference Open M with SQL creates from your Access Path Specifications. See page 18-41. |
| p | 8    | Opt       | If necessary, edit the default Row ID Specifications Open M with SQL creates from your Access Path Specifications. See page 18-43.        |
| p | 9    | Yes       | Define how to get to base table fields on the Map Data Specifications form. See page 18-45.                                               |
| p | 10   | Opt       | If necessary, define override filing code for fields that require it. See page 18-55.                                                     |
| p | 11   | Opt       | If desired, define an Index Map as conditional. See page 18-58.                                                                           |

## Field Names for Map Definition

Sometimes when you are providing a map definition, you need to enter the name of a field in the base table. When you do this, you enclose the field name in curly braces, as in {City\_Acct}.

If you can't remember the field names, it is useful to know that you can display a lookup box of legal fields by entering a space, or an open curly brace "{", and then typing a question mark, closed curly brace "}". You can also press <Search Current Table>, or enter the beginning of a field name and then press <Search Current Table>.

In the following window, the user pressed <Search Current Table> to display a lookup box of field names at the Map Data Specifications window:

-----Map Definition-----

Map Name: FavFlavors Master Map

Master Map (Y/N): Yes

Global Name: ^FLAVORS

Field(s) Used to Specify the RowID: (0/1)

People.People , FavFlavors.Piece\_Counter

-----Map Data Specifications-----

| Field         | Node | Piece                    | Delimiter | Retrieval<br>< M Code > |
|---------------|------|--------------------------|-----------|-------------------------|
| Piece_Counter |      | Piece Counter for Row ID |           |                         |
| FavFlavors    |      | FavFlavors Row ID        |           |                         |
| Flavor        |      | Person's Favorite Flavor |           |                         |
| People        |      | People Parent Reference  |           |                         |

...Map Data Specifications

Press <PF1><PF3> For Help

## Step 1: Compose SET Command to Insert Row

To accurately map a base table to an existing or planned global database on the Map Definition form, one approach is to compose a SET command that would put a row of data defined in this base table into the global database.

### **Example** of Master Map SET Command

The ^FLAVORS global has no index nodes. Therefore, you only need to create a SET command for a Master Map for the two tables which provide the relational view of ^FLAVORS: People (a parent table) and Flavors (a child table).

To put data into the People table, you write the following SET command:

```
SET ^FLAVORS(People)=" "
```

To put data in the Flavors child table, you write the following SET command:

```
SET $P(^FLAVORS(People),"^",counter)="flavor1"
```

For example:

```
SET $P(^FLAVORS("Judy"),"^",2)="Mocha"
```

### **Example** of Index Map SET Command

The ^ACCT global contains nodes which are indexes:

```
^ACCT("CITY","Butte",100)=" "
```

```
^ACCT("n",COMMUNICATIONSCONSULTANTS)=100^Communications  
Consultants
```

The generalized SET command to create the first Index Map looks like this:

```
SET ^ACCT("CITY",{City_Acct},{account_num})=" "
```

For example:

```
SET ^ACCT("CITY","Cambridge",300)=" "
```

For further details on the SET command, see “SET Commands for Master Map Structure” on page 16-14 in Chapter 16, *Default Physical Structure* .

## Step 2: Enter Map Definition Form

You define a map, whether it be the Master Map, an Index Map, or a Conditional Index Map, using the Map Definition form. The process of entering this form will differ, depending on whether or not you immediately changed to customized physical structure when you created your base table.

### If You Started Using Customized Physical Structure

If you answered "No" at the field "Use Default Physical Structure?" on the Base Table Definition master window when you first defined the base table, then Open M with SQL does not create a Master Map or any Index Maps.

#### **Procedure** To create a Master Map or Index Map:

1. Select the <Physical Structure> branching field at the Base Table Definition window.

You see the Map Definition row selection window.

2. Enter the name of your map. By convention, every table's master map is called "Data Master Map". The naming convention for index maps is the word "Index", followed by a space, then the name of the first indexed field in lowercase letters. (Default physical structure follows this with an automatically-generated index number). You may wish to include the name of the global file for the base table in your map name.
3. Press <RETURN> at the "Is this a new entry? Yes" prompt on the message line.

If this is the Master Map, press <TAB> to move the cursor to the Master Map? field, enter Y and press <RETURN>.

## If You Started Using Default Physical Structure

### **Procedure** To edit a default map:

- You see the Map Definition row selection window.

Open M with SQL Data Dictionary Guide 18-7

The highlight bar is on the Master Map, which is named Data Master Map.

4. Press <RETURN> to select the Master Map, or move the highlight bar to an Index Map.

Open M with SQL displays the Map Definition master window. The cursor is on the Map Name field.

5. Optionally, press <Erase Field> to remove the default map name and enter a descriptive name for your map.

### Examining the Map Definition Master Window

After creating or selecting a map, Open M with SQL displays the Map Definition master window.

The Map Definition master window for the Master Map of the People base table, as it appears if you had started defining the People base table using default physical structure is shown below. Note the default name, ^mdata, at the Global Name field. If you had started with customized physical structure, there would be no name in the Global Name field.

| -----Map Definition-----                                                                          |                        |
|---------------------------------------------------------------------------------------------------|------------------------|
| Map Name: People Master Map_____                                                                  | Master Map (Y/N): Yes_ |
| Global Name: ^mdata_____                                                                          |                        |
| Field(s) Used to Specify the RowID: (0/1)<br>People.People                                        |                        |
| < Access Path Specifications ><br>< Full Row Reference >                                          |                        |
| < Map Data Specifications ><br>< Row ID Specifications >                                          |                        |
| Full Row Reference: (0/1)                                                                         |                        |
| {%row} = ^mdata({})                                                                               |                        |
| -----                                                                                             |                        |
| <b>...Map Definition</b> <span style="float: right;">Press &lt;PF1&gt;&lt;PF3&gt; For Help</span> |                        |
| DELETE Row ID Specifications                                                                      | Advanced Options       |

The table on the following pages describes each field on the Map Definition master window.

**Table 18-1: Map Definition Master Window**

| Field Name                         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Map Name                           | <p>Name of the map; up to 26 alphanumeric characters, including blanks. Must be unique within the base table, but can repeat within the database.</p> <p>Give the Master Map a descriptive name, preferably one that identifies the global, or first global, to which it maps. You should develop naming conventions to use when a table is mapped to more than one global.</p>                                                                                                                                                                                                                                    |
| Master Map (Y/N)                   | <p>This field must say Yes for the Master Map. Only one map for a base table can be the Master Map. Any other map will have a default value of No in this field, which you cannot change.</p>                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Global Name                        | <p>Enter the name of the global which holds the data in this base table. The name may be an extended reference. If more than one global is used to store the data, you enter an asterisk ("*"). If the data is stored in a non-standard fashion, such as a local array, leave this field blank. See "Step 3: Specify Global Name" on page 18-11.</p>                                                                                                                                                                                                                                                               |
| Field(s) Used to Specify the RowID | <p>Output only. Shows the Row ID field, or the fields on which the Row ID is based. Open M with SQL lists the fields the Row ID is based on as specified in your Row ID field definition. See "Examining the Row ID Field Definition" on page 16-3 in Chapter 16, <i>Default Physical Structure</i>.</p> <p>On the Master Map, you can use only these fields when you define Access Path Specifications. On Index Maps, other fields in the base table may be used as well.</p>                                                                                                                                    |
| <Access Path Specifications>       | <p>This branching field opens the Access Path Specifications form, where you define the database access path Open M with SQL should use to retrieve a row of data for this table. You identify this path level by level. Each level is called an Access Level. Commonly, each level contains a subscript. Sometimes a level contains a piece or a global. Sometimes it's a customized algorithm, such as \$EXTRACT. Usually, you are defining the left side of the SET command you composed to put data into a row of this base table. See "Step 4: Define Standard Access Path Specifications" on page 18-14.</p> |
| <Full Row Reference>               | <p>This branching field opens the Full Row Reference form. Open M with SQL stores the value on this form in the variable {%row}. It uses this Full Row Reference to locate data in a base table row. Open M with SQL creates a default Full Row Reference based on the Access Path Specifications you define. You may override this value. See "Step 7: Specify Override to Full Row Reference" on page 18-41.</p>                                                                                                                                                                                                 |

**Table 18-1: Map Definition Master Window**

| Field Name                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <Map Data Specifications>    | This branching field opens the Map Data Definition form. At this form you define the location of each field in a row of the base table. Usually, you are defining the right side of the SET command you composed. See “Step 9: Define Map Data Specifications” on page 18-45.                                                                                                                                                                                                                                                                                   |
| <Row ID Specifications>      | This branching field opens the Row ID Specifications window. Open M with SQL uses the value displayed in this window to create the Row ID, given a Full Row Reference. Open M with SQL creates a default Row ID Specification from your Access Path Specifications. However, if you use M expressions as any of your Value Expressions, the Row ID Open M with SQL generates may be incorrect. Open M with SQL issues a warning and you need to check, and if necessary, edit the Row ID specification. See “Step 8: Edit Row ID Specifications” on page 18-43. |
| Full Row Reference           | Output only. Displays the current Full Row Reference, which is either the default value Open M with SQL creates or the override value you enter at the Full Row Reference form when you select the <Full Row Reference> branching field.                                                                                                                                                                                                                                                                                                                        |
| DELETE Row ID Specifications | When you define Access Path Specifications, if you enter an Access Level of type "Other", or if you enter an M expression in an Access Level's Value Expression field, the default Row ID Specifications Open M with SQL generates may be incomplete. Select this horizontal menu option to delete the existing specifications and cause Open M with SQL to regenerate the specifications. Then examine the new default value to see if it is accurate.                                                                                                         |
| Advanced Options             | This horizontal menu option lets you define filing code (INSERT, UPDATE and DELETE) for a row of data when you have used mapping features that prevent Open M with SQL from generating its own filing code. See “Step 10: Define Override Filing Code” on page 18-55.<br><br>You can also define conditional maps from this option. A Master Map may not be conditional. See “Step 11: Define an Index Map as a Conditional Map” on page 18-58.                                                                                                                 |



## Step 3: Specify Global Name

The data in a base table is located in a global, global node, or piece or part of a global node value. Sometimes, the data is located in more than one global. Sometimes, it is stored in a non-standard fashion. The way the data is stored determines the value you enter at the Global Name field on the Map Definition master window.

If you provide a global name, Open M with SQL uses it as the starting point for the Full Row Reference. Open M with SQL inserts it as the default value of the first Access Level's Data Access Expression. See “Step 4: Define Standard Access Path Specifications” on page 18-14.

### Data Stored in One Global

If the data is stored in one global, simply enter the name of that global at the Global Name field. A global name can be up to 8 alphanumeric characters in length. It cannot contain embedded blanks and the first character must be a letter or the percent symbol (%). There is case sensitivity: ^A is not equivalent to ^a.

### Extended Global Reference

If the global is located in a directory other than the directory from which the application is run, you need to use extended global reference. There are two forms of extended reference that you may use:

- n Bracket syntax, as in:
  - ^[dir]global, if the global is on the same computer
  - ^[dir,dirset]global, if the global is on another computer
- n Environment syntax, as in:
  - ^|^"^^dir"|global, if the global is on the same computer
  - ^|^"^^dirset^^dir"|global, if the global is on another computer

Only use a syntax that is supported by your underlying version of M. All versions of Open M with SQL M support bracket syntax. Versions of Open M with SQL UNIX starting with 5.1 also support the environment syntax.

### Global Name Used for Data Access Expression for Access Level 1

Open M with SQL uses this global name as the default Data Access Expression for Access Level 1. To learn how to define Access Levels to reach a row of this base table, see “Step 4: Define Standard Access Path Specifications” on page 18-14.

**Example** In the Birthdays base table, all data is stored in the ^SAMPLE global, which is located in the directory /friends on a UNIX system. The Global Name field in the

top portion of the Map Definition master window for the Birthdays base table is shown below:

| -----Map Definition-----              |                        |
|---------------------------------------|------------------------|
| Map Name: Birthdays Master Map_____   | Master Map (Y/N): Yes_ |
| Global Name: ^["/friends"]SAMPLE_____ |                        |

### Data Stored in Two or More Globals

Sometimes, due to application logic, data is stored across two or more globals. In other words, perhaps the first 10,000 rows are stored in the global ^Patients1, the next 20,000 in the global ^Patients2, and the next 20,000 in the global ^Patients3. This storage scheme is often referred to as "fragmented tables".

If this is the case in your database, enter an asterisk ("\*") in the Global Name field:

Global Name: \* \_\_\_\_\_

### Data Access Expression for Level 1 Undefined

If you use an asterisk for Global Name, Open M with SQL does not provide a default Data Access Expression for Access Level 1 in Access Path Specifications. Instead, you provide an M expression that defines the name of the global to use. See "Step 4: Define Standard Access Path Specifications" on page 18-14.

### Data Stored in Local Array

Sometimes, you may want to store your data in a local array. For example, suppose your data is stored in multiple globals and/or in a complex fashion. However, you only want to maintain one or two base tables and/or you do not want to be bothered with advanced mapping techniques. You can accomplish this by writing a routine that loads the data from the globals into a local array. You then create one base table and map it to the simple structure of that local array.

#### **Procedure** To specify global name:

1. At the Map Definition master window, move the cursor to the Global Name field.
2. If you had selected default physical structure and Open M with SQL therefore created a default Master Map, you see the default global name, ^mdata (or ^\*parent). If so, press <Erase Field> to remove the default global name.

3. Enter the name of the global where this base table's data is located, an asterisk if it is mapped to multiple globals, or leave the field blank if you are mapping to a local array.

## Step 4: Define Standard Access Path Specifications

In this step, you describe the path through your M global hierarchy that Open M with SQL will use to access a row of data in this base table. You need to define each subscript or comparable level in the M global that leads to where the base table's data is stored. These levels are called Access Levels. Open M with SQL generates the Full Row Reference from the Access Levels you define. It also attempts to generate the Row ID Specifications from your Access Level definitions.

Usually, a row of data is the value of a global node, as in the Invoices table:

```
^ACCT(account_num,invoice_num)=Order_Date:Payment_Date^Total
```

Sometimes, it is the value of that node plus one or more lower level nodes, as in the Accounts table:

```
^ACCT(account_num)=City_Acct^Name_Acct  
^ACCT(account_num, "Phone")=Phone_Acct
```

Constant subscripts, such as "Phone" in the example above, are defined in Map Data Specifications.

Sometimes, a row is a piece or part of a node value, as in the FavFlavors table:

```
^FLAVORS("people")=FavFlav^FavFlav^FavFlav
```

Sometimes, some rows are in one global and other rows are in other globals, as in COSTAR applications.

All these situations can be handled as you describe the Access Levels to reach a Full Row Reference.

There are several phases to this step, which are described in separate procedures:

- a. Enter the Access Path Specifications form

If you started with default physical structure, you need to clear the default mapping that Open M with SQL created.

Then, as you define each Access Level necessary to locate a row of data, you:

- b. Specify Data Access Expression, if the default value is not correct.
- c. Specify how to get to this Access Level, based on the Data Access Expression, by providing Access Type and Value Expression.

**Note:** If you have a complex global structure, you may need to specify Special Access Code. For instance, you may have rows stored in more than one global, or your data may be stored in a local array. See "Step 5: Specify Special Access Code" on page 18-29

to learn about locating and validating a particular Full Row Reference.

### A. Enter Access Path Specifications Form

#### **Procedure** Enter the Access Path Specifications form:

1. Press <RETURN> at the <Access Path Specifications> field on the Map Definition master window.

You see the Access Path Specifications master window. The cursor will be on the first row, on the field <Data Access Expression>. The Access Path Specification master window for the People base table is shown below. It uses the default physical structure and Open M with SQL to create a default Master Map.

```

+-----Map Definition-----+
Map Name: People Master Map_____ Master Map (Y/N): Yes_
Global Name: ^FLAVORS_____
Field(s) Used to Specify the RowID: (0/1)
People.People

+-----Access Path Specifications-----+
Access  Data Access  Access  Special
Level   Expression   Type    Value Expression   Access Code
L1      <Expression>   Sub_____ {People}_____ <Access Code>
      <Expression>   _____ <Access Code>
      <Expression>   _____ <Access Code>
      <Expression>   _____ <Access Code>
      <Expression>   _____ <Access Code>
      <Expression>   _____ <Access Code>
+-----+
...Access Path Specifications Unsaved Data      Press <PF1><PF3> For Help
  
```

Table 18-2, Access Path Specifications Master Window Fields, describes the fields in this window:

**Table 18-2: Access Path Specifications Master Window Fields**

| Field Name             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Access Level           | Output only. This column contains the name of the Access Level, which Open M with SQL assigns starting with L1. The variable {L/} refers to the value in the corresponding Value Expression column. You can use the Access Level names like field names to refer to an Access Level. You may want to do this if you provide an override Data Access Expression or provide Special Access Code for a level.                                                                                                                                                                                 |
| Data Access Expression | Press <RETURN> at this branching field to see the default Data Access Expression that Open M with SQL defines to get to the current Access Level. Open M with SQL creates this from the Data Access Expression from the prior level together with the Access Level definition for the prior level. Open M with SQL uses the Data Access Expression as the context for the current Access Level's Value Expression. In some global storage situations, you need to provide an override to Open M with SQL's default value. See "Defining an Override Data Access Expression" on page 18-19. |
| Access Type            | In this column, you enter a value to define the role of the Value Expression, which you provide in the next column. There are four Access Types: Sub (subscript), Global, Piece, or Other. The default type, if no value is entered here, is Sub.                                                                                                                                                                                                                                                                                                                                          |
| Value Expression       | Enter a value that describes how to reach this Access Level, based on the current Data Access Expression. You can enter a constant, as in "Joe Jones" or 123, a field name enclosed in curly braces, as in {Patient_Name}, or an expression including constants and/or field names.                                                                                                                                                                                                                                                                                                        |
| Special Access Code    | This branching field opens the Special Access Code form. If not all values at an Access Level correspond to rows in this base table, you can define: <ul style="list-style-type: none"> <li>■ First and/or last subscript value where data is located.</li> <li>■ Code in the NEXT subroutine that lets Open M with SQL find the next row in the table. Press the &lt;Special Code&gt; branching field.</li> <li>■ Special Data Access Variables. Press the &lt;Additional Data Variables - M Code&gt; branching field.</li> </ul>                                                         |

2. If this is a default map, delete the default mapping that appears in this window.

Follow these steps:

- a. Press <Delete Row>.

You see the following horizontal menu:

```
EXIT                                Delete ALL 'Access Specs'
```

- b. Move the highlight bar to Delete ALL 'Access Spec', and press <RETURN>.
- c. Answer *Y* at the "This option will FILE the changes. OK to delete all subscripts? No\_\_" prompt on the message line.

## **B. Examine, and If Necessary, Override, Data Access Expression**

Normally, you only need to examine the default Data Access Expression Open M with SQL provides to verify that it is accurate. You specify an override value for this variable only when the default value Open M with SQL generates is incorrect.

Usually, an Access Level has a corresponding Data Access Expression. This expression shows the data access path Open M with SQL follows to reach the current Access Level. The Data Access Expression is used as the context for the Access Type and Value Expression values you enter.

### **Data Access Expression Names**

Open M with SQL gives each Data Access Expression the name {*iD*}, where *i* is the number of the Access Level. For example, {2D} represents the Data Access Expression used to reach the second Access Level {L2}. You can use these names to refer to the value of a Data Access Expression at the following windows:

- n Data Access Expression
- n Special Access Code
- n Override Full Row Reference
- n Map Data Node
- n M Retrieval Code
- n Additional Data Access Variables

### **Default Data Access Expression**

Open M with SQL usually provides a default, or implicit, Data Access Expression, which is simply the value of the last Access Level.

**Example of Default for Access Level 1**

For Access Level {L1}, the corresponding Data Access Expression name is {1D}.

If you do not provide a global name at the Global Name field for this map, Open M with SQL provides no default Data Access Expression. In other words, the Data Access Expression is of type NULL. (See Table 18-3: Ways of Forming Data Access Expressions, on page 18-19.)

If you provide a name in the Global Name field on the Map Definition master window, Open M with SQL uses that name as the default value of {1D}. The Data Access Expression window for Access Level {L1} for the People table is shown below:

-----Map Definition-----

Map Name: Recipes Master Map

Master Map (Y/N): Yes\_

Global Name: ^RECIPES

Field(s) Used to Specify the RowID: (0/1)  
recipes.recipes

-----Data Access Expression-----

Implicit Reference at this level:  
{1D} = ^RECIPES

Override Reference:  
{1D} = You may specify here an override reference to the above:

...Data Access Expressions

Press <PF1><PF3> For Help

**Types of Data Access Expressions**

There are four ways you can form a Data Access Expression. The form of the Data Access Expression limits the Access Type you can enter for the Access



Level. These ways of forming a Data Access Expression are described in the following table:

**Table 18-3: Ways of Forming Data Access Expressions**

| Form                        | Description and Interaction with Corresponding Value Expression                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Global Reference            | <p>If the first character of the Data Access Expression is "^", it is a global reference, such as ^global(Sub1,Sub2).</p> <p>The Data Access Expression must be of this type in order for the Access Level to have a Value Expression of Access Type "Sub".</p> <p>If the previous Access Level is of Access Type "Sub", meaning subscript, then the default Data Access Expression for the current level is a global reference, with its final subscript equal to the Value Expression defined at that prior level.</p> |
| Piece                       | <p>If the Data Access Expression of the previously referenced Access Level is a global reference or a piece and the Access Type of that previous Level is "Piece", the default Data Access Expression is in the form of a \$PIECE expression, such as:</p> <p>\$PIECE(^FLAVORS({Name}),"^",{Piece_Counter}))</p>                                                                                                                                                                                                         |
| Other non-NULL M expression | <p>The Data Access Expression will default to this type if:</p> <ul style="list-style-type: none"> <li>The previously referenced Access Level was of Access Type "Piece".</li> <li>You enter an override Data Access Expression and don't begin it with a "^", indicating a global reference.</li> </ul>                                                                                                                                                                                                                 |
| NULL                        | <p>The default Data Access Expression is NULL only if:</p> <ul style="list-style-type: none"> <li>The previously referenced Access Level is of Access Type "Other"</li> <li>It is Access Level 1 and you did not specify a global name at the Global Name field for this map</li> </ul> <p>If the Data Access Expression is NULL (the default is NULL and no override is given), the current Access Level must be of Access Type "Other".</p>                                                                            |

### Defining a Local Array

The local array name must begin with a % sign.

### Defining an Override Data Access Expression

If the implicit Data Access Expression Open M with SQL provides does not accurately define an Access Level, as it won't if you entered an asterisk at Global Name or left that field blank, you can define an override Data Access Expression.

An override Data Access Expression may only be of two types:

- n Global Reference
- n Other Non-NULL M expression

You can refer to values from a prior Access Level in your override Data Access Expression. For instance, Data Access Expression {2D} can only refer to variables from the first Access Level: {L1}, {1D} and {1Di}.

You cannot use field names, such as {Birthdays.Name}, in your Data Access Expression.

**Procedure** To examine and provide an override value for the Data Access Expression:

1. Press <RETURN> at the <Expression> branching field on the Access Path Specifications master window.

You see the Data Access Expression window.

2. Examine the default value Open M with SQL provides under the prompt "Implicit Reference at this level:".
3. If necessary, enter an override reference, using the allowed variables in your M expression.

**Example** You need to define an override Data Access Expression if you define a child table and its data is stored in a global different from that of the parent table. For instance, let's say you have a table, Patients, mapped to the global ^A(Patient\_ID). You create a child table from Patients, called "Office\_Visits", mapped to ^B(Patient\_ID,Visit\_Date).

When you map the Access Path Specifications for the Office\_Visits table, you need to provide an override value for {L2}. The table "Access Path Specifications for Office\_Visits" shows the default and override values for Data Access Expression, Access Type and Value Expression in {L1} and {L2}:

**Table 18-4: Access Path Specifications for Office\_Visits**

| Access Level | Default Data Access Expression | Override Data Access Expression | Access Type | Value Expression |
|--------------|--------------------------------|---------------------------------|-------------|------------------|
| {L1}         | ^A                             |                                 | Sub         | {Patient_ID}     |
| {L2}         | ^A({L1})                       | ^B({L1})                        | Sub         | {Visit_Date}     |

The result is the following Full Row Reference for the Office\_Visits child table:

`{%row}=^B( {L1} )`

## **C. Specify How to Reach Current Access Level**

In this step, you define the Access Type and the Value Expression that describe how to get to the current Access Level. Open M with SQL uses this information

in conjunction with the Access Level's Data Access Expression to create the description of this Access Level. It uses this as the default value of the next Data Access Expression. If this is the last Access Level, this value is the Full Row Reference.

### Access Type

This field defines the role of the Value Expression. There are four Access Types, described in the Table 18-5: Access Types, on page 18-21.

With the exception of Access Type "Other", and sometimes "Global", Open M with SQL automatically provides default values for the Special Code window fields "Invalid Values", "NEXT subroutine code", or the next Access Level's Data Access Expression. See "Define Special Access Code to Validate Row" on page 18-30.

**Table 18-5: Access Types**

| Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Sub    | A global subscript, which Open M with SQL adds hierarchically to the current Data Access Expression to form the Data Access Expression for the next level, or for the final Full Row Reference if this is the last Access Level. Default value if you leave this field blank.                                                                                                                                                                                                                                                                                                                                       |
| Piece  | Open M with SQL applies the \$PIECE function to the current Data Access Expression using the value of a field you create in the base table and which you name in the Value Expression column. Open M with SQL uses this field as a piece counter, automatically initializing and updating its value as it retrieves rows of data.<br><br>When you specify "Piece", a small window appears where you enter the delimiter that separates the rows in the global node. The delimiter can be a constant enclosed in quotes, such as "^", or an expression such as \$C(1) [which represents the vertical bar character]. |
| Global | The Value Expression resolves to a global name.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Other  | None of the above. Open M with SQL provides no defaults for the Special Access Code fields "Invalid Values", "NEXT subroutine code", or the next Access Level's Data Access Expression, so you need to supply them yourself. You must specify this type if there is no Data Access Expression for the Access Level (the Data Access Expression is NULL).                                                                                                                                                                                                                                                            |

The table "Location of Full Row and Access Type" describes how the location of a full row affects the Access Types you define:

**Table 18-6: Location of Full Row and Access Type**

| Location of Full Row                           | Access Type                                                                                                                                                  |
|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Full Row is Global Node Value or Subscript     | All your Access Levels will be of type "Sub", unless your data is in more than one global, in which case the Access Type for {L1} is "Global".               |
| Full Row is a Delimited Piece of Node Value    | The final Access Level will be of Access Type "Piece".                                                                                                       |
| Full Row is an Undelimited Piece of Node Value | The final Access Level will be of Access Type "Other" and you will use an M expression, which might include a function such as \$EXTRACT, to retrieve a row. |

### Value Expression

This field defines the value of the current Access Level. You may enter a:

- n Constant, which can be:
  - A string, such as "333 West 34th Street"
  - A number, such as 123
- n Field name enclosed in curly braces, such as:

{FavFlavors}

or, if the table is linked via an implicit join:

{People.Name}

In the Master Map, you may only use the Row ID field, or fields on which the Row ID is based, in the Value Expression. In Index Maps, you may use any field in the base table.

**Note:** This is the only window in Access Path Specifications where you may use field names.

- n Expression which includes constants and/or field names and/or M variables, such as:

1000\*{counter}+offset

or

\$\$ALPHAUP ( {Name} )

The table "Location of Full Row and Value Expression" describes how the location of a full row affects the Value Expressions you define:

**Table 18-7: Location of Full Row and Value Expression**

| Location of Full Row                           | Value Expression                                                                                                                                                                                       |
|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Full Row is Global Node Value or Subscript     | The final Access Level will include a subscript value, which can be a constant or a Row ID field name in curly braces by itself or in an M expression.                                                 |
| Full Row is a Delimited Piece of Node Value    | The final Access Level will be the name of the piece counter field you defined in your base table field definitions. Open M with SQL uses this field to manage looping through rows in the base table. |
| Full Row is an Undelimited Piece of Node Value | The final Access Level will be of Access Type "Other" and you will use an M expression, which might include a function such as \$EXTRACT, to retrieve a row.                                           |

### Functions in Value Expression and Index Maps

If your application frequently issues a SELECT statement which uses a Collating Function, such as \$\$ALPHAUP or \$\$UPPER, to retrieve data, it is a good idea to create an Index Map based on that function. Include the actual value of the field in the Index Map so Open M with SQL does not have to access the Master Map as well to retrieve the actual field value. You do this by including the field acted on by the function, in the Index Map's Access Path Specifications, and then including the actual field in Map Data Specifications. See "Collation Sequence Affects Name and Text Data Types" on page 6-28 in Chapter 6, *Defining Base Table Fields*.

**Table 18-8: Collating Function Index Map Syntax**

| Collating Function | Index Map Syntax | Index Map Example                                                     |
|--------------------|------------------|-----------------------------------------------------------------------|
| ALPHAUP            | \$\$ALPHAUP      | <code>^mdata(TableName,col, \$\$ALPHAUP{indexfld1},{RowID})=""</code> |
| EXACT              | [default]        | <code>^mdata(TableName,col, {indexfld1},{RowID})=""</code>            |
| Minus              | -                | <code>^mdata(TableName,col,-{indexfld1},{RowID})=""</code>            |
| Plus               | +                | <code>^mdata(TableName,col, +{indexfld1},{RowID})=""</code>           |

Table 18-8: Collating Function Index Map Syntax

| Collating Function | Index Map Syntax | Index Map Example                                                  |
|--------------------|------------------|--------------------------------------------------------------------|
| Space              | " _              | <code>^mdata(TableName,col,"_{indexfld1},{RowID})=""</code>        |
| UPPER              | \$\$UPPER        | <code>^mdata(TableName,col,\$\$UPPER{indexfld1},{RowID})=""</code> |

**Example of Function in Index Map**

Suppose your application or lookup frequently uses the following SQL query:

```
SELECT Account_Name
FROM Accounts
WHERE $$ALPHAUP(Account_Name)=%edit(3)
```

Remember that the ^ACCT global structure looked like this:

```
^ACCT( "n" , $$ALPHAUP {Account_Name} ) = {account_num} ^ {Account
_Name }
```

{Account\_Name} is stored in ALPHAUP format in the second subscript position of the index. you define the Index Map with the following Access Path Specifications.

-----Map Definition-----

Map Name: Accounts Index Map\_\_\_\_\_

Master Map (Y/N): No\_\_

Global Name: ^ACCT\_\_\_\_\_

Field(s) Used to Specify the RowID: (0/1)

Accounts.Accounts

-----Access Path Specifications-----

| Access Level | Data Expression | Access Type | Value Expression               | Special Access Code |
|--------------|-----------------|-------------|--------------------------------|---------------------|
| L1           | <Expression>    | Sub         | "n"                            | <Access Code>       |
| L2           | <Expression>    | Sub         | \$\$ALPHAUP ( {Account_Name} ) | <Access Code>       |
|              | <Expression>    |             |                                | <Access Code>       |
|              | <Expression>    |             |                                | <Access Code>       |
|              | <Expression>    |             |                                | <Access Code>       |
|              | <Expression>    |             |                                | <Access Code>       |

...Access Path Specifications Unsaved Data

Press <PF1><PF3> For Help

If in the field definition for {Account\_Name} you specified using the \$\$ALPHAUP function for lookups, then the search for the matching name will be very efficient, as no transformation of the value in the database will be needed.

---

---

Examples of Access Path Specifications

Birthdays Base Table

The Birthdays base table is accessed via one Access Level, as shown below. It is based on the Data Access Expression, "^SAMPLE". The Access Type is sub-script. The value is the field {Name}, which is the Row ID of this table.

-----Map Definition-----

Map Name: Birthdays Master Map\_\_\_\_\_Master Map (Y/N): Yes\_  
Global Name: ^SAMPLE\_\_\_\_\_  
Field(s) Used to Specify the RowID: (0/1)  
Sample.Name

-----Access Path Specifications-----

| Access Level | Data Access Expression | Access Type | Value Expression (Name) | Special Access Code |
|--------------|------------------------|-------------|-------------------------|---------------------|
| L1           | <Expression>           | Sub_____    | _____                   | <Access Code>       |
|              | <Expression>           | _____       | _____                   | <Access Code>       |
|              | <Expression>           | _____       | _____                   | <Access Code>       |
|              | <Expression>           | _____       | _____                   | <Access Code>       |
|              | <Expression>           | _____       | _____                   | <Access Code>       |
|              | <Expression>           | _____       | _____                   | <Access Code>       |

...Access Path SpecificationsInquiry ModePress <PF1><PF3> For Help

FavFlavors Base Table

Remember a row of this base table is a piece of the node value. The first Access Level is the same as that for the People table. If you look at the Data Access Expression for {L2}, you see the value of {2D} is ^FLAVORS({People}).



The next window shows the process of creating the second Access Level for the FavFlavors base table. We define it as Access Type "Piece" and then enter the delimiter:

| -----Map Definition-----             |                 |             |                      |                        |  |  |
|--------------------------------------|-----------------|-------------|----------------------|------------------------|--|--|
| Map Name: FavFlavors Master Map_____ |                 |             |                      | Master Map (Y/N): Yes_ |  |  |
| Global Name: ^FLAVORS_____           |                 |             |                      |                        |  |  |
| -----Access Path Specifications----- |                 |             |                      |                        |  |  |
| Access Level                         | Data Expression | Access Type | Value Expression     | Special Access Code    |  |  |
| L1                                   | <Expression>    | Sub         | {People.People}_____ | <Access Code>          |  |  |
| L2                                   | <Expression>    | Piece       | _____                | <Access Code>          |  |  |
| Piece Delimiter: "^"_____            |                 |             |                      |                        |  |  |
| <Expression> _____ <Access Code>     |                 |             |                      |                        |  |  |

...Piece Delimiter      Unsaved Data      Press <PF1><PF3> For Help

For the Value Expression, we enter the name of the field we entered as Piece\_Counter:

| -----Map Definition-----             |                 |             |                      |                        |  |  |
|--------------------------------------|-----------------|-------------|----------------------|------------------------|--|--|
| Map Name: FavFlavors Master Map_____ |                 |             |                      | Master Map (Y/N): Yes_ |  |  |
| Global Name: ^FLAVORS_____           |                 |             |                      |                        |  |  |
| -----Access Path Specifications----- |                 |             |                      |                        |  |  |
| Access Level                         | Data Expression | Access Type | Value Expression     | Special Access Code    |  |  |
| L1                                   | <Expression>    | Sub         | {People.People}_____ | <Access Code>          |  |  |
| L2                                   | <Expression>    | Piece       | {Piece_Counter}_____ | <Access Code>          |  |  |
|                                      | <Expression>    | _____       | _____                | <Access Code>          |  |  |
|                                      | <Expression>    | _____       | _____                | <Access Code>          |  |  |
|                                      | <Expression>    | _____       | _____                | <Access Code>          |  |  |
|                                      | <Expression>    | _____       | _____                | <Access Code>          |  |  |

...Access Path Specifications      Press <PF1><PF3> For Help

## Delete Access Level Specifications

You can delete all Access Level specifications or only those for the last Access Level. If you started defining your base table using default physical structure, you delete all the default Access Level specifications Open M with SQL creates before entering your own specifications.

**Procedure** To delete the specifications for the last Access Level:

1. Move the cursor so it is on the last Access Level you defined.
2. Press <Delete Row>.

You see the following horizontal menu:

```
EXIT Delete Last Access LevelDelete ALL Access Level  
Specifications
```

3. Move the highlight bar to the horizontal menu item "Delete Last Access Level" and press <RETURN>.

**Procedure** To delete all Access Level Specifications:

1. Do Steps 1 and 2 from the procedure "To delete the specifications for the last Access Level:" on page 18-28.
2. Move the highlight bar to the horizontal menu option "Delete ALL Access Level Specifications" and press <RETURN>.

## Step 5: Specify Special Access Code

For most global structures, the standard Access Path Specifications you define allows Open M with SQL to accurately access and validate a row of data. However, for some global structures, not all values at an Access Level correspond to valid paths to a row of data in the base table. In such cases, the default methods Open M with SQL uses based on the standard Access Path Specifications may not work and you need to define additional Special Access Code.

In this section, we first explain the default methods Open M with SQL uses to access and validate a row of data. Then we explain how you can add your own validation and accessing code to meet the requirements of your existing M database.

### How Open M with SQL Accesses and Validates a Row

In general, in a data retrieval query such as a SELECT statement, Open M with SQL accesses a row in one of the following two modes:

- n Single value access
- n Loop access

#### Single Value Access

In single value access, Open M with SQL knows the Row ID of the desired row. Open M with SQL uses the Row ID Specifications to plug the Row ID field or fields into the Value Expression of its corresponding Access Level. Open M with SQL then performs its default test for validity of that Full Row Reference: IF \$DATA(Full Row Reference)'=0. If that test proves TRUE, then Open M with SQL considers the Full Row Reference to be valid.

**Example** In the FavFlavors table, the Full Row Reference is:

```
^FLAVORS( { People } )
```

Open M with SQL uses single value access if you issue a SELECT statement such as the one below, where the Row ID is provided:

```
SELECT Flavor
FROM FavFlavors
WHERE { People }="Judy"
```

If \$DATA(^FLAVORS("Judy")) is non-zero, the Full Row Reference is valid. The field Flavor is actually in the child table, FavFlavors. Open M with SQL will return the value of the first row in that table, which is the first favorite flavor of Judy.

### Loop Access

In loop access, Open M with SQL does not know the Row ID of the desired row. Rather, it searches through all rows, starting with the first valid Full Row Reference, until it either finds a row which meets the condition specified in the query or until it reaches a NULL value. It locates the next row by using the \$ORDER function on "Sub" Access Types and a counter increment on "Piece" Access Types. Open M with SQL therefore locates rows in collating sequence according to subscript.

**Example** In the FavFlavors table, the Full Row Reference is:

```
$PIECE(^FLAVORS({People}),"^",Piece_Counter)
```

Open M with SQL will use loop access if you issue the following command, since the Row ID of FavFlavors is not provided:

```
SELECT People
FROM FavFlavors
WHERE FavFlav="Vanilla"
```

It will return the value of the People field for the first row in FavFlavors where the field FavFlav contains the value "Vanilla". It loops through the rows by doing the following two steps:

1. {L1} is of type "Sub", with the value being {People}. Open M with SQL uses:

```
$O(^FLAVORS({People}))
```

2. {L2} is of type "Piece" with the value being {Piece\_Counter}. Open M with SQL initializes {Piece\_Counter} to 0 and then uses:

```
SET {Piece_Counter}={Piece_Counter}+1
SET value=$P(^FLAVORS({People}),"^",{Piece_Counter})
```

### Define Special Access Code to Validate Row

If your global structure is complex, the default single access validation condition (\$DATA of the Full Row Reference) may validate and the default loop code (\$ORDER of the Full Row Reference) may point to a Full Row Reference which does not actually point to a row in the base table. In this case, you must describe certain types of Access Level limitations at the Special Access Code form (and its nested forms) which Open M with SQL checks to determine if it has a valid row. These limitations include:

- n First Value and Last Value subscript values for loop access.
- n IF conditions for single value access.
- n NEXT subroutine code for loop access.

## Types of Special Access Code

For each Access Level, you can enter Special Access Code on four different windows in the Special Access Code form:

| -----Map Definition-----                                                                        |                        |
|-------------------------------------------------------------------------------------------------|------------------------|
| Map Name: Data Master Map_____                                                                  | Master Map (Y/N): Yes_ |
| -----Special Access Code-----                                                                   |                        |
| First Value: _____                                                                              | Last Value: _____      |
| <Special Code>                                                                                  |                        |
| < Additional Data Variables - M Code>                                                           |                        |
| <div> <div>...Special Access Code</div> <div>Press &lt;PF1&gt;&lt;PF3&gt; For Help</div> </div> |                        |

At the Special Access Code window, you can specify:

- n First and last valid Access Level values

Open M with SQL uses these first and last valid values to provide default values for the following two fields, each found on a separate window.

- n Special Code for Invalid Access Level values when single access retrieval is used, which is used in addition to \$DATA(). From the Special Access Code window, press the <Special Code> action field to display the Special Code window. See page 18-33.
- n Custom code for the NEXT subroutine, for which Open M with SQL ordinarily uses \$ORDER on the Access Level value. Press the <NEXT Subroutine> branching field from the Special Code window. See page 18-35.

The following sections discuss conditions under which you can employ these features to allow Open M with SQL to locate valid Access Level values in your M global.

- n Additional Data Access Variables

From the Special Access Code window you can select the <Additional Data Variables - M Code> branching field to define variables you can access at various

mapping windows. See “Step 6: Specify Additional Data Access Variables” on page 18-39.

Range of Subscript Values Not Part of Base Table

When an Access Level is of type "Sub", sometimes not all values of that subscript lead to rows in the base table. If there is a pattern to the value change, you can often define the First and Last valid subscript values in the Special Access Code form.

If the valid values are all numbers, simply set the type of that field to Number in its field definition. If the valid values are all uppercase alphabetic, then set the First Value to "A" and the Last Value to "ZZZ", or a string of length equal to the maximum length of the subscript.

Default Invalid Values

Open M with SQL automatically sets up default Invalid Conditions for an Access Level value based on the First and Last Values you entered. It also enhances the NEXT subroutine code to reflect these invalid Access Level values.

Procedure To define first and last valid Access Level value:

- 1. Press <RETURN> at the branching field <Special Code> on the Access Path Specifications master window.

You see the Special Access Code window.

-----Map Definition-----

Map Name: Account Master Map

Master Map (Y/N): Yes\_

-----Special Access Code-----

First Value:

Last Value:

<Special Code>

< Additional Data Variables - M Code>

...Special Access Code

Press <PF1><PF3> For Help

2. Enter the first value for the subscript at that level Open M with SQL should use when it retrieves rows of data in a loop.
3. Enter the last value for the subscript Open M with SQL should use when it retrieves rows of data in a loop.

## **Single Value Access and Invalid Values**

For times when single access retrieval mode is used, and therefore the NEXT subroutine is not employed, Open M with SQL uses the default or override code specified at the Special Code window to test the supplied Access Level value. It uses this code in addition to testing the value of \$DATA on the Full Row Reference to make sure it is non-zero.

### **Default Invalid Values**

If you entered First and Last Values at the Special Access Code window, Open M with SQL automatically generates default invalid values for you.

### **Override Conditions Replace Invalid Values**

The default code Open M with SQL creates, and which is displayed in the Invalid Values field, is in effect only if you do not enter code in the Override Conditions field. Therefore, if you enter first and last values but wish to enter other invalid conditions as well, you must include the Invalid Values code Open M with SQL creates based on first and last values when you enter your code in the Override Conditions field.

### **Procedure To examine or override default Invalid Values code:**

1. Press <RETURN> at the branching field <Special Code> on the Special Subscript window.

You see the Special Code window:

```
+-----Map Definition-----+
|
| Map Name: Account Master Map_____ Master Map (Y/N): Yes_
| +-----Special Access Code-----+
| +-----Special Code-----+
|
| Invalid Values:
|   The following is used if nothing is entered below:
|   Value Invalid: (0/)
|       IF
|       OR IF
|
| Override Conditions:
|   You may override the above Invalid Values by using the following:
|   Value Invalid: (1/0)
|       IF _____
|       OR IF _____
|       OR IF _____
|       OR IF _____
|
|                                     <NEXT Subroutine>
|
+-----+
|...Special Code                                     Press <PF1><PF3> For Help|
+-----+
```

2. If desired, enter invalid conditions in the Override Conditions portion of the screen.

Remember, the default IF conditions are only in effect if there is no value in the Override Conditions: field.

### NEXT Subroutine to Access Valid Access Level Values

When Open M with SQL does a loop access of an Access Level, it uses the NEXT subroutine to find the next valid value.

#### Using \$ORDER for Access Levels Which are Global References

If \$ORDER is used, you first set the value of the Access Level to NULL. Open M with SQL does this automatically if the default NEXT subroutine is used.

#### Default NEXT Subroutine

The default subroutine is used only if you do not enter an Override Subroutine.



### **Override NEXT Subroutine**

When entering an override NEXT subroutine, note the following:

- n The pseudo-tag "NEXT" is the tag of the first line of the subroutine. Within the subroutine, this tag may be used with a GOTO command to mean:  

```
"this value is invalid, go look for the next one"
```
- n This pseudo-tag is replaced by an actual tag during code generation. Other lines in the subroutine may not have tags.
- n You may reference this and previous subscripts by using the level name enclosed in curly braces, as in {L1}, {L2}.
- n The value of the current Access Level should be set by an M SET of the level name, enclosed in {}. For example, `SET {L1}=$ORDER(%local({L1}))`.
- n This subroutine is called with a DO statement, and is exited with a QUIT statement. QUIT commands may appear anywhere within the subroutine.
- n Upon exiting, the Access Level value must be one of the following:
  - Constant
  - M expression including Row ID fields
  - a valid subscript value
  - NULL

**Note:** A NULL value is used to indicate that there are no more valid values in this subscript level.

### **Override NEXT Subroutine Does Not Include First and Last Value Conditions**

The default subroutine Open M with SQL creates is in effect only if you do not enter code in the Override Conditions field. Therefore, if you enter first and last values but wish to enter an override NEXT subroutine, you must include the code Open M with SQL creates based on first and last values in your override subroutine.

### **Reactivating Default NEXT Subroutine**

The default subroutine can be reactivated by the user at any time by setting the "Override" field to NULL. The system will never automatically modify or delete what the user enters in this window.

#### **Procedure To examine or provide override code for the NEXT subroutine:**

1. Press <RETURN> at the <NEXT Subroutine> branching field on the Special Code window.

You see the NEXT Subroutine window. The top portion of the window shows the default NEXT subroutine Open M with SQL generates from the "Last Value" entry and the applicable "Invalid Values" entry.

-----Map Definition-----

Map Name: Account Master Map

Master Map (Y/N): Yes\_

-----Special Access Code-----

-----Special Code-----

-----NEXT Subroutine-----

The following is used if no Override Subroutine is entered:

Subroutine Code: (0/2)

NEXT s {L1}=\$o(^ACCT({L1}))

q:{L1}=""

QUIT

Override Subroutine:

Subroutine Code: (1/0)

NEXT

QUIT

...NEXT Subroutine

Press <PF1><PF3> For Help

Example of Using Special Code

The rows in the Accounts base table are located at the first and second subscript levels in the ^ACCT global, with the first subscript level being the Full Row Reference.

However, only some of the first subscript values lead to rows in the Master Map. The other subscripts lead to rows in the Index Maps. Since the valid subscript values are all numbers, the easiest way to be certain to get valid values is to define the {account\_num} field as a Number data type.

However, you can also specify first and last values, as shown below:

```

+-----Map Definition-----+
| Map Name: Account Master Map_____ Master Map (Y/N): Yes_ |
+-----Special Access Code-----+
|
| First Value: 1_____ Last Value: "A"_____ |
|
| <Special Code> |
|
| < Additional Data Variables - M Code> |
|
+-----+
|
| ...Special Access Code      Unsaved Data      Press <PF1><PF3> For Help |
|
+-----+

```

Open M with SQL generates default Invalid Conditions code, based on the First and Last values you enter.

```

+-----Map Definition-----+
| Map Name: Account Master Map_____ Master Map (Y/N): Yes_ |
+-----Special Access Code-----+
+-----Special Code-----+
|
| Invalid Values: |
|   The following is used if nothing is entered below: |
|   Value Invalid: (0/2) |
|       IF 1>{L1},{L1}'=+{L1} |
|       OR IF {L1}] "A",{L1}'=+{L1} |
|
| Override Conditions: |
|   You may override the above Invalid Values by using the following: |
|   Value Invalid: (1/0) |
|       IF _____ |
|       OR IF _____ |
|       OR IF _____ |
|       OR IF _____ |
|
| <NEXT Subroutine> |
+-----+
|
| ...Special Code      Unsaved Data      Press <PF1><PF3> For Help |
|
+-----+

```

Open M with SQL also edits the NEXT subroutine to reflect the invalid conditions implied by the values you enter in First Value and Last Value.

-----Map Definition-----

Map Name: Account Master Map

Master Map (Y/N): Yes\_

-----Special Access Code-----

-----Special Code-----

-----NEXT Subroutine-----

The following is used if no Override Subroutine is entered:

Subroutine Code: (1/4)

NEXT s {L1}=\$o(^ACCT({L1}))  
i {L1}] "A",{L1}'=+{L1} s {L1}="" ; From 'Last Value'  
q:{L1}=""  
QUIT

Override Subroutine:

Subroutine Code: (0/)

NEXT \_\_\_\_\_  
\_\_\_\_\_  
QUIT

...NEXT Subroutine      Unsaved Data      Press <PF1><PF3> For Help

## Step 6: Specify Additional Data Access Variables

You rarely need to define Additional Data Access Variables. However, this feature allows you to handle an M global structure where:

- n Fields in a row are located in multiple storage locations, necessitating a more complex Full Row Reference. You can define Additional Data Access Variables which you use to define the Full Row Reference.
- n Data in a row exists at a level of a global higher than the Full Row Reference.

### Names of Additional Data Access Variables

You reference a Data Access Variable as  $\{iDd\}$ , where  $\{iD\}$  is the Data Access Expression reference and  $d$  is the Additional Data Access Variable expression number. For example,  $\{2D1\}$  represents the first Data Access Variable expression of Access Level 2.

### Where You Can Use Data Access Variables

Data Access Variables can be used in the following locations:

- n Override Full Row Reference
- n Map Data Node
- n M Retrieval Code
- n Special Access Code
- n Subsequent Data Access Variable code

### **Procedure** To define Additional Data Access Variables:

1. Press <RETURN> at the <Special Code> branching field at the Access Path Specifications master window for the level where you wish to define one or more special variables.

You see the <Special Code> master window.

2. Press <RETURN> at the <Additional Data Variables - M Code> branching field.

You see the Additional Data Access Variables - M Code window:

-----Map Definition-----

Map Name: Account Master MapMaster Map (Y/N): Yes

-----Special Access Code-----

First Value: 1Last Value: "A"

<Special Code>

-----Additional Data Access Variables - M Code-----

M Code for Additional Data Variables:

|     |       |                                                        |
|-----|-------|--------------------------------------------------------|
| Var | Var   | You may have additional Data Access Variables by using |
| #   | Name  | SET {iDd}=... in the following code lines:             |
| 1   | {iD1} | SET {iD1}=^INVOICE({L1},{L2})                          |
| 2   | {iD2} | SET {iD2}=^ITEM(\$p({iD1},"^",2))                      |

...Additional Data Access VariablesPress <PF1><PF3> For Help

The table "Additional Data Access Variables Window" describes the fields on this form:

Table 18-9: Additional Data Access Variables Window

| Field Name | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Var #      | Enter the number of the variable.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Var Name   | Output only. Open M with SQL displays the name of this variable. Use this name to access the value of this variable elsewhere in Map Definition.                                                                                                                                                                                                                                                                                                                                                                                                        |
| SET {iDd}  | Enter one or more lines of M code that sets the value of the Data Access Variable. If your code includes multiple paths, be sure each path provides a value for the Data Access Variable. You can include the name of routine, Access Levels ({L}), Data Access Expressions, {iD}, or previously defined Data Access Variables, {iDd}. You may not use field names. Open M with SQL sets the value of these variables, one by one, following the d order, immediately after setting the value of the Access Level, {L}, with which they are associated. |

3. Enter code for as many Data Access Variables as you wish to define at this Access Level.

## Step 7: Specify Override to Full Row Reference

Open M with SQL uses the Full Row Reference to access a row of data in the table. It appends expressions you define in the Map Data window to locate each field in the base table.

### Default Full Row Reference

Open M with SQL creates a default Full Row Reference from the result of the final Access Level you define in Access Path Specifications. This is equivalent to the Data Access Expression for the level past the last Access Level for which you provide a Value Expression or Special Access Code.

### {%row} Represents Full Row Reference

The special variable {%row} contains the value of the Full Row Reference. It is the implicit value used for the location of fields you specify in Map Data Specifications.

You can use this variable to represent the value of the Full Row Reference when defining:

- n Special Access Code
- n Map Data M Retrieval Code

### Override Full Row Reference

If your data is stored in more than one location, you may need to provide an override to the Full Row Reference. You can use Data Access Expression variables, {iD}, or Data Access Variables, {iDd}, in the M expression you use to define the override value.

#### **Procedure** To examine or override the default Full Row Reference:

1. Select the <Full Row Reference> branching field at the Map Definition master

You see the Full Row Reference window:

-----Map Definition-----

Map Name: FavFlavors Master Map\_\_\_\_\_

Master Map (Y/N): Yes\_

Global Name: ^FLAVORS\_\_\_\_\_

Field(s) Used to Specify the RowID: (0/1)  
People.People , FavFlavors.Piece\_Counter

-----Full Row Reference-----

Implicit Full Row Reference: (0/1)  
{%row} = \$p(\$g(^FLAVORS({L1})), "^", {L2})

Override Full Row Reference:  
{%row} = \_\_\_\_\_

...Full Row Reference

Press <PF1><PF3> For Help

- 2. If necessary, enter an override value at the Override Full Row Reference field.



## Step 8: Edit Row ID Specifications

Once Open M with SQL locates a row in loop access, it must calculate the value of the Row ID field or fields. It uses the Row ID Specifications to accomplish this. Row ID Specifications indicate how to determine the value of each field in the Row ID, based in the values of each Access Level you define.

If you use an M expression as the Value Expression at the Access Path Specifications window, Open M with SQL may not be able to generate accurate Row ID specifications.

### **Procedure** To examine or override the Row ID Specifications:

1. If you see the message "Row ID Specification no longer valid" on the message line, follow these steps:

- a. Press <Go to Bottom Menu> at the Map Definition master window.
- b. Press <RETURN> at the option Delete Row ID Specifications.

You see the following prompt on the message line:

Delete & Regenerate Row ID Specifications? \_\_\_\_

- c. Enter Yes to have Open M with SQL recreate the Row ID Specifications.
2. Select the <Row ID Specifications> branching field at the Map Definition master window to see the default specifications Open M with SQL generated:

| -----Map Definition-----             |             |                  |
|--------------------------------------|-------------|------------------|
| -----Access Path Specifications----- |             |                  |
| Access Level                         | Access Type | Value Expression |
| L1                                   | Sub         | {People.People}  |
| L2                                   | Piece       | {Piece_Counter}  |
|                                      |             |                  |
|                                      |             |                  |
|                                      |             |                  |
|                                      |             |                  |

| -----Calculate Row ID Given: Access Levels and Fields----- |            |
|------------------------------------------------------------|------------|
| Field                                                      | Expression |
| People.People                                              | {L1}       |
| Piece_Counter                                              | {L2}       |
|                                                            |            |
|                                                            |            |
|                                                            |            |
|                                                            |            |

|                       |                           |
|-----------------------|---------------------------|
| ...Row ID Calculation | Press <PF1><PF3> For Help |
|-----------------------|---------------------------|

The table below describes the fields on the Row ID Specifications window:

**Table 18-10: Row ID Specifications Window**

| Field Name | Description                                                                                                                                                                                                                                                                |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Field      | Output Only. Contains the name of the Row ID, or the fields on which the Row ID field is based.                                                                                                                                                                            |
| Expression | Contains the M expression used to calculate the value of the corresponding Row ID field. You can use Access Level, Data Expression or Additional Data Access Level variables, as well as field names defined in the Map Data window in curly braces, in your M expression. |

3. If necessary, delete existing specifications and enter new M code that accurately describes how to calculate the value of each field in the Row ID.

## Step 9: Define Map Data Specifications

As described in the Checklist for Creating a Relational View, InterSystems recommends that you first map one data field and then test your Access Path Specifications with a query. Once you are sure you have correctly defined how to reach a row of data in Access Path Specifications, you are ready to define how to find the remaining fields in the row.

You define data field locations on the Map Data Specifications form.

### Two Ways to Define Field's Location

You can define a field's location in one of two ways:

- n **Standard Specifications**

If the field is located at or below the Full Row Reference, and it is either a node value, or piece of the node value, then you define Node, and/or Piece and Delimiter.

- n **Retrieval M Code**

If the field is not located at or below the Full Row Reference, or if it must be extracted from part of a node value, you define M Retrieval Code. You also need to define your own Override Filing Code. See “Step 10: Define Override Filing Code” on page 18-55.

### Standard Specifications

The following sections describe the standard fields on the Map Data Specifications screen: Node, Piece and Delimiter. The remaining sections describe using M Retrieval Code.

#### Node Specification

The Node field specifies the node contains a specified field in the base table. Its value usually represents additional levels of subscripting beneath the Full Row Reference. These subscripts must be constant or literal values. They may not be variable field values. This is because a variable field value must be part of the Access Path Specifications.

In addition to describing additional subscripts, you may also use any previously defined non-NULL `{iD}`, `{iDd}`, or `{%row}` reference as the entry in the Node field. Piece/Delimiter entries may optionally be used in conjunction with a Node entry of this type, to further identify the location of the field. Leaving the Node field blank is equivalent to using `{%row}` as the Node entry. Note that use of any `{ }` reference in the Node field means that the Node is not a descendent of the Full

Row Reference, but an entirely independent location from which to retrieve the field.

**Caution:** *If any {iD} or {iDd} reference is specified as the Node entry, Open M with SQL will not automatically generate filing code for that field. If such code is necessary, you may enter it either through the "Filing Override" feature (see "Step 10: Define Override Filing Code" on page 18-55), or by using pre-filing and post-filing base table triggers (see Chapter 11, Base Table Triggers ).*

The table "Possible Node Field Values" shows how the semantics of Node can designate different access strategies:

**Table 18-11: Possible Node Field Values**

| Node Value | Meaning                                                            |
|------------|--------------------------------------------------------------------|
| Blank      | Use the Full Row Reference.                                        |
| Literal    | Use a descendent of the Full Row Reference.                        |
| +n or -n   | Use an offset from the lowest subscript in the Full Row Reference. |
| {%row}     | Use the Full Row Reference (same as leaving Node blank).           |
| {iD}       | Use the specified Data Access Expression.                          |
| {iDd}      | Use the specified Additional Data Access Variable.                 |

**Note:** *Node may only have a literal value or +n or -n if the Full Row Reference ({%row}) takes the form of a global reference that may be further subscripted. The same holds for a multi-line field stored in nodes, which also requires additional subscripting.*

### Multi-Line Field

If your data is in one of the two standard Open M with SQL multi-line storage formats, you define the field as multi-line at the Field Definition window, and then you simply name the node which contains its value in Map Data Specifications.

### Piece and Delimiter Specification

If a field is a piece of a node value, you need to specify which piece it is, and the delimiter that separates the pieces.

## Examples of Standard Map Data Specifications

### Field is Value of Full Row Reference

If the field is represented by the entire value of the Full Row Reference, then you simply enter the field name in the Field column. Enter nothing in the other columns.

**Example** The value of a row in the FavFlavors table is the value of the entire row (which, since the row is a piece of a node, happens to be a piece of a node value.)

```

+-----Map Definition-----+
|
| Map Name: FavFlavors Master Map_____ Master Map (Y/N): Yes_
|
| Global Name: ^FLAVORS_____
|
| Field(s) Used to Specify the RowID: (0/1)
| People.People , FavFlavors.Piece_Counter
|
+-----Map Data Specifications-----+
|
| Field      Node      Piece      Delimiter      Retrieval
| FavFlav_____      _____      _____      _____      < M Code >
| _____      _____      _____      _____      < M Code >
| _____      _____      _____      _____      < M Code >
| _____      _____      _____      _____      < M Code >
| _____      _____      _____      _____      < M Code >
| _____      _____      _____      _____      < M Code >
| _____      _____      _____      _____      < M Code >
|
+-----+
|
| ..Map Data Specifications                                     Press <PF1><PF3> For Help
|

```

### Field is Multi-Line

In this example, the table testmulti contains two fields of data type text:

{Allinone} is a multi-line field using All in One Node storage mode:

```
^A(1,2)=Line3^Line4
```

{Onepernode} is a multi-line field using One Per Node storage mode:

```
^A(1,1,0)=2
```

```
^A(1,1,1)=Line1
```

```
^A(1,1,2)=Line2
```

The Map Data Specifications for these fields are shown below:)

-----Map Definition-----

Map Name: testmulti Master Map\_\_\_\_\_

Master Map (Y/N): Yes\_

Global Name: ^A\_\_\_\_\_

Field(s) Used to Specify the RowID: (0/1)

testmulti.testmulti

-----Map Data Specifications-----

| Field           | Node   | Piece | Delimiter | Retrieval  |
|-----------------|--------|-------|-----------|------------|
| allinone_____   | 2_____ | _____ | _____     | < M Code > |
| onepernode_____ | 1_____ | _____ | _____     | < M Code > |
| _____           | _____  | _____ | _____     | < M Code > |
| _____           | _____  | _____ | _____     | < M Code > |
| _____           | _____  | _____ | _____     | < M Code > |
| _____           | _____  | _____ | _____     | < M Code > |
| _____           | _____  | _____ | _____     | < M Code > |

...Map Data Specifications

Press <PF1><PF3> For Help

Fields are Pieces of Node Value

In many cases, a node contains the values of more than one field, separated by piece delimiters. If this is the case, when you get to the Access Level where you are defining that structure, enter the piece number in the Piece column and the delimiter that separates the pieces in the Delimiter column.

**Example** Two fields in the Accounts table are pieces of the Full Row Reference. The third field is the full value of a node one subscript level down from the Full Row Reference. The Map Data Specifications window for this scenario is shown below:

-----Map Definition-----

Map Name: Accounts Master Map\_\_\_\_\_

Master Map (Y/N): Yes\_

Global Name: ^ACCT\_\_\_\_\_

Field(s) Used to Specify the RowID: (0/1)

Accounts.Accounts

-----Map Data Specifications-----

| Field           | Node         | Piece  | Delimiter | Retrieval  |
|-----------------|--------------|--------|-----------|------------|
| City_Acct_____  | _____        | 1_____ | "^"_____  | < M Code > |
| Name_Acct_____  | _____        | 2_____ | "^"_____  | < M Code > |
| Phone_Acct_____ | "Phone"_____ | _____  | _____     | < M Code > |
| _____           | _____        | _____  | _____     | < M Code > |
| _____           | _____        | _____  | _____     | < M Code > |
| _____           | _____        | _____  | _____     | < M Code > |
| _____           | _____        | _____  | _____     | < M Code > |

...Map Data Specifications

Press <PF1><PF3> For Help

### Field is Piece of a Piece

If a field is a piece of a piece, you manage that by putting down two piece numbers in the piece column, and two delimiters in the Delimiter column.

**Example** The Invoices table contains 2 fields which are a piece of a piece, {Invoice\_Date} and {Payment\_Date}. The structure of the node looks like this:

```
^ACCT(account_num,invoice_num)=Invoice_Date:Payment_Date^Total
```

The Map Data Specifications window for the fields in the Invoices table is shown below:

| -----Map Definition-----                                                                 |       |                        |                           |            |
|------------------------------------------------------------------------------------------|-------|------------------------|---------------------------|------------|
| Map Name: Invoices Master Map_____                                                       |       | Master Map (Y/N): Yes_ |                           |            |
| Global Name: ^ACCT_____                                                                  |       |                        |                           |            |
| Field(s) Used to Specify the RowID: (0/1)<br>Accounts.Accounts , Invoices.Invoice_Number |       |                        |                           |            |
| -----Map Data Specifications-----                                                        |       |                        |                           |            |
| Field                                                                                    | Node  | Piece                  | Delimiter                 | Retrieval  |
| Invoice_Date_____                                                                        | _____ | 1,1_____               | "^",": "_____             | < M Code > |
| Payment_Date_____                                                                        | _____ | 1,2_____               | "^",": "_____             | < M Code > |
| Total_____                                                                               | _____ | 2_____                 | "^",": "_____             | < M Code > |
| _____                                                                                    | _____ | _____                  | _____                     | < M Code > |
| _____                                                                                    | _____ | _____                  | _____                     | < M Code > |
| _____                                                                                    | _____ | _____                  | _____                     | < M Code > |
| _____                                                                                    | _____ | _____                  | _____                     | < M Code > |
| -----                                                                                    |       |                        |                           |            |
| ...Map Data Specifications Inquiry Mode                                                  |       |                        | Press <PF1><PF3> For Help |            |

### Field is Located in Node Below Full Row Reference

If the field is located at a subscript level below the Full Row Reference, then you enter a value in the Node field. List all the additional subscripts beyond those included in the Full Row Reference, separated by commas.

### Field in Node Which is Offset of Full Row Reference Node

Sometimes, a field is located in a node which is at the same level as the Full Row Reference.

In a case like this, you need to do the following:

- n Enter Special Access Code to indicate which values are invalid when Open M with SQL retrieve a row.
- n Use an offset value to specify the node which contains the field at the Map Data window.

**Example** The global ^G contains one field, City, which is at a node one level below the Full Row Reference node. It also contains two fields, SS# and Age, which are in nodes which are offsets from the Full Row Reference.

```

^G(10)="Jones,Judy"
  ^G(10,"CITY")="Boston"
  ^G(11)=SS#
  ^G(12,"AGE")=25
^G(20)="Hancock,John"
  ^G(20,"CITY")="Cambridge"
  ^G(21)=SS#
  ^G(22,"AGE")=43
  
```

The Map Data Specifications window for the fields in this global is shown below:

| -----Map Definition-----                                                                                         |           |                        |           |            |
|------------------------------------------------------------------------------------------------------------------|-----------|------------------------|-----------|------------|
| Map Name: Employees Master Map_____                                                                              |           | Master Map (Y/N): Yes_ |           |            |
| Global Name: ^G_____                                                                                             |           |                        |           |            |
| Field(s) Used to Specify the RowID: (0/1)<br>Employees.Employee_Num                                              |           |                        |           |            |
| -----Map Data Specifications-----                                                                                |           |                        |           |            |
| Field                                                                                                            | Node      | Piece                  | Delimiter | Retrieval  |
| Name                                                                                                             |           |                        |           | < M Code > |
| City                                                                                                             | "CITY"    |                        |           | < M Code > |
| SSnum                                                                                                            | +1        |                        |           | < M Code > |
| Age                                                                                                              | +2, "AGE" |                        |           | < M Code > |
|                                                                                                                  |           |                        |           | < M Code > |
|                                                                                                                  |           |                        |           | < M Code > |
|                                                                                                                  |           |                        |           | < M Code > |
| ...Map Data Specifications Inquiry Mode <span style="float: right;">Press &lt;PF1&gt;&lt;PF3&gt; For Help</span> |           |                        |           |            |

## Retrieval M Code

You don't normally need to enter Retrieval M Code. However, if a field is located at a node level that is not at or below the Full Row Reference, if it is stored in another global, or if it is a concatenation of information from several different locations, you define how to locate it at the Retrieval M Code form.

If you need to use Retrieval M Code, then you leave the Node, Piece and Delimiter fields blank. If you enter a value in any of those fields, Open M with SQL will not allow you to enter the Retrieval M Code form.

**Example** Assume you have the following global structure:

```

^Global(Sub)=ABCDEFGHJIJ
  
```



Sub is the Row ID. Field 1 is the first four characters, field 2 is the second three characters and field 3 is the last 3 characters of the node value. You would obtain these values with the following M function:

```
field_1=$EXTRACT(^Global(Sub),1,4)
field_2=$EXTRACT(^Global(Sub),5,7)
field_3=$EXTRACT(^Global(Sub),8,10)
```

### Enter Own Filing Code

If you do define M Retrieval Code for a field, you must also provide override filing code for that field, since default filing code Open M with SQL provides is always based on the Full Row Reference.

### **Procedure** To map data specifications:

1. Press <RETURN> at the <Map Data> branching field.  
You see the Map Data window.
2. Enter the name of the field or press ? to see a list of fields in the table from which you can select the field you want.
3. If the field is located at or below the Full Row Reference, enter Node and/or Piece and Delimiter values and go to Step 6.
4. If the field is not located at or below the Full Row Reference, move the cursor to the <M Code> branching field and press <RETURN>.

You see the Map Data Field Retrieval Code window:

|                                                                                                              |                        |
|--------------------------------------------------------------------------------------------------------------|------------------------|
| -----Map Definition-----                                                                                     |                        |
| Map Name: Accounts Master Map_____                                                                           | Master Map (Y/N): Yes_ |
| Global Name: ^ACCT_____                                                                                      |                        |
| Field(s) Used to Specify the RowID: (0/1)<br>account.account                                                 |                        |
| +-----Map Data Specifications-----+                                                                          |                        |
| +-----Map Data Field Retrieval Code-----+                                                                    |                        |
| You may use M code to retrieve this field's value by using<br>SET {*}=... in the following code lines: (1/0) |                        |
| _____<br>_____<br>_____<br>_____                                                                             |                        |
| +-----Map Data Field Retrieval Code-----+                                                                    |                        |
| ...Map Data Field Retrieval Code                                                                             |                        |
| Press <PF1><PF3> For Help                                                                                    |                        |

The table Map Data Field Retrieval Code Window describes the field in this window:

**Table 18-12: Map Data Field Retrieval Code Window**

| Field Name | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SET {*}    | <p>Enter M code which describes the location of the field. Be certain all paths in your M code provide an M SET command which defines the location of the field. Reference this field as {*}. Open M with SQL will replace the asterisk with the field name automatically.</p> <p>You can use any previously described:</p> <ul style="list-style-type: none"><li>n Access Levels {Li}</li><li>n Data Access Expressions {iD}</li><li>n Additional Data Access Variables {iDd}</li></ul> <p>and the {%row} variable in your code. You may not reference any field name other than the one you are SETting.</p> |

5. Repeat Steps 2 through 4 until you have defined the location of all fields in the map.

## Examples of Map Data Specifications

### **Example** of Field as Value of Full Row Reference

The value of the Full Row Reference of the FavFlavors table is the value of its Flavor field. Remember the structure of the underlying ^FLAVORS global:

```
^FLAVORS( "Art" )=Vanilla^Chocolate^Coffee
```

The Map Data Specifications window for this base table is shown below. The Access Path Specifications window for this base table is shown in the “Examples of Access Path Specifications” on page 18-26.

```

+-----Map Definition-----+
Map Name: FavFlavors Master Map_____ Master Map (Y/N): Yes_
Global Name: ^FLAVORS_____
Field(s) Used to Specify the RowID: (0/1)
People.People , FavFlavors.Piece_Counter
+-----Map Data Specifications-----+
Field      Node      Piece      Delimiter      Retrieval
FavFlavors_____
< M Code >
< M Code >
< M Code >
< M Code >
< M Code >
< M Code >
< M Code >
+-----+
...Map Data Specifications                      Press <PF1><PF3> For Help

```

### Example of Field in Node Level Below Full Row Reference

In the Account table, the fields Address and Phone are located one subscript level down.

```

^ACCT(100)=Butte^Communication Consultants
^ACCT(100,"Phone")=222-3333

```

The Map Data Specifications window entries are shown below:

```

+-----Map Definition-----+
Map Name: Accounts Master Map_____ Master Map (Y/N): Yes_
Global Name: ^ACCT_____
Field(s) Used to Specify the RowID: (0/1)
Accounts.Accounts
+-----Map Data Specifications-----+
Field      Node      Piece      Delimiter      Retrieval
City_Acct_____ 1_____
Name_Acct_____ 2_____ " ^ "_____
Phone_Acct_____ "Phone"_____
< M Code >
< M Code >
< M Code >
< M Code >
< M Code >
< M Code >
< M Code >
+-----+
...Map Data Specifications                      Press <PF1><PF3> For Help

```

**Example of Fields Which Are a Piece Within a Piece**

The Invoice table contains two date fields which are pieces of a piece. Remember the global structure of the underlying ^ACCT global:

```
^ACCT(100,1000)=55740:55759^1000.00
```

The first date is the invoice date, the second date is the payment receipt date, and the final piece is the total amount of the invoice.

The Map Data Specifications window entries are shown below:

-----Map Definition-----

Map Name: Invoices Master Map\_\_\_\_\_Master Map (Y/N): Yes\_

Global Name: ^ACCT\_\_\_\_\_

Field(s) Used to Specify the RowID: (0/1)  
Accounts.Accounts , Invoices.Invoice\_Number

-----Map Data Specifications-----

| Field        | Node | Piece | Delimiter | Retrieval  |
|--------------|------|-------|-----------|------------|
| Total        |      | 2     | "^"       | < M Code > |
| Invoice_Date |      | 1,1   | "^",":"   | < M Code > |
| Payment_Date |      | 1,2   | "^",":"   | < M Code > |
|              |      |       |           | < M Code > |
|              |      |       |           | < M Code > |
|              |      |       |           | < M Code > |

...Map Data SpecificationsPress <PF1><PF3> For Help

## Step 10: Define Override Filing Code

Open M with SQL defines default filing code for a field only if { %row } is:

- n A simple global reference (Type 1) or
- n The result of one or more "Piece" levels of a simple global reference (Type 2) and the map is the Master Map.

Even if the above conditions are met, you need to define a field's filing code if:

- n You define a field's location in the Map Data Specifications form by using Retrieval M Code.
- n You use {iD} or {iDd} variables in the Node specification for a field's location at the Map Data Specifications window.

### Two Ways to Define Override Filing Code

You can define override filing code in one of two ways:

- n **For all fields**  
Use the Override Filing Code form accessed via the Advanced Options horizontal menu choice at the Map Definition master window. See the procedure below.
- n **For selected fields**  
Use base table triggers to handle the filing of fields for which Open M with SQL did not generate filing code due to the conditions described above. See Chapter 11, *Base Table Triggers*, to learn how to create such triggers.

### Referencing Fields in Filing Code

The M code, and any routines it may call, may not reference any { } variables. It may reference field values in the row being filed in accordance with the following InterSystems filing conventions using the filing arrays %data() and %edit():

- n %data(1) is the RowID value,
- n %data(c) is the most up-to-date value for a field with column number c,
- n %edit(c) is defined only if the value of field c is changing, in which case it holds the field's old value.

Note also that:

- n In case of DELETE, %edit holds the old value of each field, while all %data(c) (except for %data(1)) are set to NULL.
- n Multi-line fields are stored using the usual local array format, as follows:

```
%data(c)=n
```

with the actual values stored as follows:

```
%data(c,1), ..., %data(c,n)
```

- n If internal-to-external conversion code is defined for a field, then its value will take the following format:

```
<internal> $c(1) <external>
```

**Procedure** To override the default filing code for all fields:

1. Press <Go to Bottom Menu> at the Map Definition master window.
2. Move the highlight bar to Advanced Options and press <RETURN>.

You see the Advanced Options window:

```

+-----Map Definition-----+
|
| Map Name: People Master Map_____ Master Map (Y/N): Yes_
|
| Global Name: ^FLAVORS_____
|
| Field(s) Used to Specify the RowID: (0/1)
| People.People
|
+-----Advanced Options-----+
|
| M code to replace the generated filing code.
| Code unspecified will be replaced by system generated code.
|
| INSERT: _____
| UPDATE: _____
| DELETE: _____
|
|                                     < Conditional Map >
|
+-----+
|...Advanced Options                                     Press <PF1><PF3> For Help

```

The table Filing Code Fields describes the fields on this screen.

**Table 18-13: Filling Code Fields**

| Field Name | Description                                                                                                                                                                                                                                                                                                                                                                             |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| INSERT     | Enter a line of M code, up to 200 characters in length. You must define how to file each field in the base table, since entries here suppress any filing code Open M with SQL generates for any field in the table.<br><br>Normally, you will use a SET command, or a DO command which calls a filing routine. The M code, or any routines it calls, may not contain any {} references. |
| UPDATE     |                                                                                                                                                                                                                                                                                                                                                                                         |
| DELETE     |                                                                                                                                                                                                                                                                                                                                                                                         |

3. Enter a line of M code at the INSERT field.

For your convenience, Open M with SQL automatically propagates that code to the UPDATE and DELETE fields.

4. If necessary, edit the code at the UPDATE and DELETE fields.

Remember, you must provide code for each type of filing.

## Step 11: Define an Index Map as a Conditional Map

A Conditional Map is an Index Map that contains rows only for a subset of rows within a base table. A specified condition determines which rows are included in the map. In addition, it may contain only a subset of fields from the base table, rather than all fields. This is useful if you frequently access a subset of rows and only need the data in certain fields for your query.

### **Procedure** To make an Index Map conditional:

1. At the Map Definition master window, press <Go to Bottom Menu>.
2. Move the highlight bar to the <Advanced Options> branching field and press <RETURN>.

You see the Advanced Options window:.

| -----Map Definition-----                                                                                    |                        |
|-------------------------------------------------------------------------------------------------------------|------------------------|
| Map Name: Accounts Index Map_____                                                                           | Master Map (Y/N): No__ |
| Global Name: ^ACCT_____                                                                                     |                        |
| Field(s) Used to Specify the RowID: (0/1)<br>Accounts.Accounts                                              |                        |
| -----Advanced Options-----                                                                                  |                        |
| M code to replace the generated filing code.<br>Code unspecified will be replaced by system generated code. |                        |
| INSERT: _____                                                                                               |                        |
| UPDATE: _____                                                                                               |                        |
| DELETE: _____                                                                                               |                        |
| < Conditional Map >                                                                                         |                        |
| -----                                                                                                       |                        |
| ...Advanced Options                                                                                         |                        |
| Press <PF1><PF3> For Help                                                                                   |                        |

3. Move the cursor to the <Conditional Map> branching field and press <RETURN>.



You see the Conditional Map Definition window:

```

+-----Map Definition-----+
| Map Name: Accounts Index Map_____ Master Map (Y/N): No__ |
| Global Name: ^ACCT_____ |
+-----Conditional Map Definition-----+
| Conditional Map (Y/N): No__ |
| Fields for Conditional Updates: _____ |
| (0/) _____ |
| Set Map IF _____ |
| Percentage of Rows in this Map: _____ |
| Use Conditional Map when comparison against host variables? No__ |
+-----+
| ...Conditional Map Definition | Press <PF1><PF3> For Help |

```

The table below describes the fields on the Conditional Map window:

**Table 18-14: Conditional Map Window**

| Field Name                                                 | Description                                                                                                                                                                                                                                                                                                  |
|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Conditional Map (Y/N)                                      | NO inactivates all other options in this window except Use Conditional Map when comparison against host variables. YES activates all window options.                                                                                                                                                         |
| Fields for Conditional Updates                             | Enter fields to be included in each row of Conditional Map.                                                                                                                                                                                                                                                  |
| Set Map IF                                                 | Enter condition(s) a row must meet to be included in Conditional Map. May contain multiple conditions separated by commas, as in the Standard M IF command. Can include field names in curly braces.                                                                                                         |
| Percentage of Rows in this Map                             | Enter an estimate of the percentage of rows from the Master Map which will be included in the Conditional Map. This estimate will be used by Open M with SQL to optimize the use of this map.                                                                                                                |
| Use Conditional Map when comparison against host variables | This parameter allows you to specify if Conditional Maps are valid index maps for a WHERE clause comparing to a host variable.<br><br>YES specifies that Conditional Maps may be used. The host variable must <i>never</i> be null ("").<br><br>NO specifies do not use Conditional Maps. NO is the default. |

- 4. At the Fields for Conditional Updates field, enter the names of the fields from the base table which you want to include in the Conditional Map.
- 5. At the Set Map IF field, enter the condition(s) for determining row inclusion in the map.
- 6. At the Percentage of Rows in this Map field, enter the percentage of rows you estimate will satisfy this condition.

**Example** In an employees application where you often deal separately with retired employees, you might want to make a Conditional Map which includes records for retired employees only:

-----Map Definition-----

Map Name: Employees Index Map\_\_\_\_\_Master Map (Y/N): No\_\_

Global Name: ^Employees\_\_\_\_\_

-----Conditional Map Definition-----

Conditional Map (Y/N): Yes\_

Fields for Conditional Updates: {Name}\_\_\_\_\_  
(1/2) {Pension\_Monthly}\_\_\_\_\_

Set Map IF {Retired}="Y"\_\_\_\_\_

Percentage of Rows in this Map: 10\_\_\_\_\_

Use Conditional Map when comparison against host variables? No\_\_

...Conditional Map DefinitionPress <PF1><PF3> For Help



---

## ***Special Topics***

---

Chapter 19

### ***The FileMan Interface***

Chapter 20

### ***Importing Data Definitions***

---





---

## The FileMan Interface

---

FileMan is an M-based public domain software package that was developed and is distributed and maintained by the U.S. Department of Veteran Affairs. Inter-Systems provides an automated interface from VA FileMan to Open M with SQL.

This chapter describes how to use the FileMan Interface to connect FileMan data to Open M with SQL. Specifically, it covers the following topics:

|                                                             |            |
|-------------------------------------------------------------|------------|
| n Overview of the FileMan Interface                         | page 19-2  |
| n Accessing the FileMan Interface                           | page 19-8  |
| n Creating the FileMan-Open M Link                          | page 19-10 |
| n Extending, Updating, and Deleting the FileMan-Open M Link | page 19-13 |
| n Generating Views of FileMan Files                         | page 19-23 |
| n FileMan-Open M with SQL Cross Reference Report            | page 19-25 |
| n Conversion Error Log                                      | page 19-27 |
| n Using the FileMan Interface with Open M with SQL PDP      | page 19-29 |
| n Limitations to the FileMan Interface                      | page 19-30 |
| n FileMan Interface Questions and Answers                   | page 19-32 |

## Overview of the FileMan Interface

To use the FileMan Interface, you must be running Version 18.1 or higher of the VA FileMan software.

The FileMan Interface is a one-way link that creates one or more Open M with SQL data dictionary specifications from a FileMan data dictionary specification. Once created, the Open M with SQL data dictionary coexists with the FileMan data dictionary but as a distinct and separate entity. The FileMan-Open M with SQL link does not remain active beyond the initial connection, meaning it does not automatically update changes made to one data dictionary into the other. You can, however, manually relink FileMan files to Open M with SQL tables in order to update the link with changes made to the FileMan data dictionary. You can never link Open M with SQL tables back to FileMan files.

When you connect a FileMan file to Open M with SQL, the FileMan Interface converts the file into an Open M with SQL relational table, which is then available for data insert, update, delete, or query using forms created in the Form Generator, queries created using the Open M with SQL query facilities, and reports created in M/PACT.

### Database Normalization

A FileMan file is composed of a main file which may have any number of fields and sub-files. Fields with multiple values are stored in a sub-file under the main file. For example, a hospital patient file may have a visit sub-file for multiple visit dates associated with a single patient. Other fields may have one logical occurrence but have multiple lines, e.g., a word processing field used to hold comments. Such hierarchical relationships are not permissible within a relational database.

The FileMan Interface uses normalization to connect FileMan files that have hierarchical logical relationships to two-dimensional Open M with SQL tables. This action converts all one-to-many relationships to characteristic relationships or parent-child links. In FileMan terminology, all sub-files and word processing fields are converted to child tables within Open M with SQL.

### Conversion Logic

When you create the FileMan-Open M with SQL link, the FileMan Interface uses the information in the FileMan data dictionary specification to create one or more relational base tables in Open M with SQL. The interface routines are recursive. Each time Open M with SQL finds a sub-file or word processing field in a field specification, the interface suspends processing of the current file and creates a child table from the sub-file specification. If the sub-file in turn has its own sub-files, the interface also creates their child tables before control returns to the current file. Recursive logic is also used if a field is a pointer to another file. In such



instances, the interface suspends processing of the current file in order to convert the file to which the field in the original table points.

## Identifiers

Some FileMan identifiers for files, sub-files and fields do not conform to naming conventions required by Open M with SQL. The interface handles unacceptable identifiers in the following fashion:

- n If an identifier does not start with an alphabetic character, a single "f" character is placed as a prefix to the identifier.
- n In FileMan field names, all punctuation other than the underscore character "\_" is removed. All spaces are converted to the underscore character. (See FileMan Preferences, below.)
- n In FileMan field numbers, the decimal point (.) character is converted to the underscore character by default. Because underscore characters are eliminated from base table names before checking for uniqueness, you may want to translate the decimal point character into another alphanumeric character to ensure unique base table names. (See FileMan Preferences, below.)
- n If the identifier is longer than the maximum allowed length (40 characters), vowels are removed from the identifier starting at the end and working backwards until an acceptable length is reached. Once all vowels have been stripped, the identifier is truncated if it exceeds the allowable length. (See FileMan Preferences, below.)
- n If the identifier is not unique or is an SQL reserved word, the FileMan field number is appended to the end of the identifier to make it acceptable. Optionally, you can append the FileMan field number to all base table names. (See FileMan Preferences, below.)

For example, the FileMan file:

```
MAIN HOSPITAL (NORTH WING) POST-SURGERY PATIENTS TABLE
```

would be converted to an Open M with SQL table name as follows:

```
MAIN_HOSPITL_NRTH_WNG_PSTSRGRY_PTNTS_TBL
```

FileMan Preferences

The FileMan Preferences window, accessed from the FileMan Interface main menu, gives you more specific control over FileMan-to-base table translation mapping:

-----FileMan Preferences-----

Map Word Processing fields as Multi-line?

Enable mapping?

Append Fileman File Number to Table Name?

< Data Type Options >

File Name -> Base Table Name Character Translations:

Translate From: To:

----- Name Mapping Conventions -----

Subfile Prefix

Eliminate Duplicate Words?

< Mandatory Abbreviations >

< Additional Abbreviations >

Disable Mapping of files/fields starting with

FileMan Preferences

Press <PF1><PF3> For Help

Enter Yes or No.

Table 19-1: FileMan Preferences Options

|                                                     |                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Map Word Processing fields as Multi-line?           | YES/NO field.                                                                                                                                                                                                                                                                                                                                                                                                 |
| Enable Mapping?                                     | Required YES/NO field.                                                                                                                                                                                                                                                                                                                                                                                        |
| Append FileMan File Number to Table Name?           | YES/NO field.<br>YES appends the FileMan file number to all base table names.<br>NO only appends the FileMan file number when it is required to create a unique base table name.                                                                                                                                                                                                                              |
| <Data Type Options>                                 | This action field displays the Data Type Options window. This window allows you to specify the following:<br><br>Does the installed version of FileMan allow you to convert seconds? YES/NO field for the Time data type.<br><br>Force Length of Date Fields to be X Characters Long. Specify a number value for X. Length is used for all Date data type fields, and overrides all checks during conversion. |
| File Name -> Base Table Name Character Translations | Specify a list of characters in the From and To fields. Every instance of each character in the From field is translated to the corresponding To character (see example below).                                                                                                                                                                                                                               |

**Example** To eliminate vowels and underscores from Base Table names, define File Name->Base Table Name Character Translations as follows:

|                                                                                                          |
|----------------------------------------------------------------------------------------------------------|
| File Name -> Base Table Name Character Translations:<br>Translate From: <b>A E I O U a e i o u _</b> To: |
|----------------------------------------------------------------------------------------------------------|

A generated Fileman Base Table named 'PATIENT\_10000' would be translated to 'PTNT10000'. All vowels and underscores have been translated to an empty string ('null').

**Table 19-2: Name Mapping Conventions Options**

|                                               |                                                                                                                                                                                                                                                                                 |
|-----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Subfile Prefix                                | Enter the prefix for the subfile. A subfile prefix may be up to 10 characters long, may contain only letters, numbers, and the underscore character, and must begin with a letter.                                                                                              |
| Eliminate Duplicate Words                     | Specify YES to eliminate duplicate words when mapping FileMan names onto Open M with SQL base table, view, and field names. Specify NO to retain duplicate words.                                                                                                               |
| <Mandatory Abbreviations>                     | Press <RETURN> at this action field to define mandatory abbreviations for mapping FileMan names onto Open M with SQL base table, view, and field names.                                                                                                                         |
| <Additional Abbreviations>                    | Additional abbreviations are automatically applied during name mapping when the use of Mandatory Abbreviations creates a name that is still too long. Press <RETURN> at this action field to define additional abbreviations to be used along with the mandatory abbreviations. |
| Disable mapping of files/fields starting with | Specify the initial characters (up to 15 characters) of the names of fields or files that are not to be mapped to base tables or base table fields.                                                                                                                             |

## File Specifications

For main files, the interface derives the Open M with SQL table name from the FileMan file name. This is done according to the logic for converting identifiers discussed previously. If a description field exists, the first line of the field is converted to the base table description. If no description exists, the description is copied from the table name.

For sub-files, the base table name is composed of "SUB" followed by the name of the sub-file. The description is created from the description of the parent file field pointing to the sub-file, followed by the parent file number.

## Field Specifications

The FileMan Interface derives the Open M with SQL field name from the FileMan field name, according to the logic for converting identifiers. The field description is derived from the FileMan field title.

The following table lists FileMan data types and shows the Open M with SQL data types to which the FileMan Interface converts them:

**Table 19-3: FileMan Data Types**

| <b>FileMan Data Type</b> | <b>Open M with SQL Data Type</b>                                   |
|--------------------------|--------------------------------------------------------------------|
| Date                     | Text (with conversion/validation code)                             |
| Computed Date            | Number (with internal FileMan date flag)                           |
| Time                     | Text (with conversion/validation code)                             |
| Number                   | Number                                                             |
| Computed                 | Text                                                               |
| Pointer                  | Designative Reference                                              |
| Variable Pointer         | Main field - Text<br>Each designated table - Designative Reference |
| Sets of codes            | Multiple Choice                                                    |
| All others               | Text                                                               |

All converted FileMan fields are placed on a base table master map, except for computed fields and fixed-length fields. Fixed-length fields are maintained through post-filing triggers.

### Variable Pointer Fields

In Open M with SQL, variable pointer fields are represented by multiple fields. There is a main field that is mapped and holds the variable pointer field value. The main field is defined as a Text field. There is also one field for each table designated by the variable pointer field. These fields are represented as Designative Reference computed fields that are always equal to the calculation and are based on the main field. Their field names are formed by a three digit number (starting with 001 and incremented by one for each field) appended to the first 14 characters of the main field name. For a given value of the main field, the other fields are computed to null, except for the field which corresponds to the table pointed by that value. The code to compute them will always be updated to reflect the current designated tables.

### **Field Mapping**

Open M with SQL maps all FileMan fields as nodes and pieces of a global.

The only exception to this rule are fields defined with the function \$EXTRACT. Open M with SQL maps these fields via base table triggers.

### **Indices**

Open M with SQL uses all FileMan indices. It treats only regular indices as M with SQL index maps. Post-filing triggers maintain other index types. Because of this, only regular indices are candidates for database searches and retrievals. The other index types are maintained for database integrity, but are not used for searches and/or retrievals.

### **FileMan Interface Does Not Link ScreenMan Applications**

The FileMan Interface links only FileMan files to Open M with SQL. It does not link applications built using the ScreenMan application generator.

If you have such applications, you can link the underlying FileMan files to Open M with SQL, then recreate forms on top of the relational tables using the Form Generator component of Open M with SQL.

# Accessing the FileMan Interface

The FileMan Interface is automatically distributed with every Open M with SQL Developer license. You can only access the FileMan Interface when the FileMan ^DIC and ^DD globals exist in your current directory. The ^DIC global holds file names, and the ^DD global holds fields, sub-files, and pointers.

When these globals are present, Open M with SQL automatically loads the File-Man Interface

**Procedure To access the FileMan Interface:**

- 1. Enter Open M with SQL.
- 2. From the Open M with SQL main menu, select the Data Dictionary option.

When the FileMan ^DIC and ^DD globals are present in the current directory, the Data Dictionary menu appears as follows:

```

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA OPEN M Developer
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
UAAAAAAAAAAAAData DictionaryAAAAAAAAAA
3
3 Base Table Definition 3
3 View Definition 3
3 FileMan Interface 3
3 Compile a Table 3
3 Copy a Base Table Field 3
3 Copy a View 3
3 Change View's Starting Table 3
3 Recreate all Default Structure Maps 3
3 Populate Index Maps for a Table 3
3 Reports on Data Dictionary 3
3
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAU

Wednesday Feb 19, 1997 Directory: /dbmsl/morrow/
Copyright (c) 1993 A InterSystems Corporation

Data Dictionary 12:02PM Press <PF1><PF3> For Help
Press <RETURN> to link FileMan files to Open M Developer
```

When the FileMan globals are not present in the current directory, the Data Dictionary menu suppresses display of the third option, FileMan Interface.

- 3. From the Data Dictionary menu, select the FileMan Interface option.

You will see the FileMan Interface menu, as shown below

```

----- OPEN M Developer -----
+-----FileMan Interface-----+
|
| FileMan-Open M Developer Link
| Link All FileMan Files To Developer
| Generate Views of FileMan Files
| FileMan Preferences
| Test Name Conversion
| Print FileMan-Developer Cross Ref.
| Print Conversion Error Log
| Purge Conversion Error Log
|
+-----+

```

Directory: /dbms1/morrow/  
 Wednesday Feb 19, 1997      Copyright (c) 1993 - InterSystems Corporation

```

FileMan Interface 12:04PM
Press <PF1><PF3> For Help
Link a FileMan file and its sub-files to Developer

```

This menu provides a full range of capabilities for linking your FileMan application to Open M with SQL, including a Link Maintenance facility for extending, updating, and deleting pieces of the link, and a View Generation facility for joining the tables that comprise a main file.

It also provides several FileMan Interface reporting options, including a FileMan-Open M with SQL Cross Reference report and a Conversion Error Log report.

## Creating the FileMan-Open M Link

The first step in using the FileMan Interface is to create the FileMan—Open M link. This link maps FileMan files to Open M with SQL relational tables.

There are two ways to create the FileMan—Open M link:

1. You can create the link for all FileMan files within a specified range.
2. You can create the link one FileMan file at a time.

Method #1 automatically links all FileMan files within the specified range to Open M with SQL. Depending on the size of the range you specify and the complexity of your application, this can be a time-consuming process. However, it is the most efficient way to link a large number of files.

Method #2 is a more conservative approach. It lets you build the link on a file-by-file basis, where you individually select each file that you want to connect to Open M with SQL. This capability is most useful when you want to extend the link by adding new files after the original link is already in place.

When you link a file, Open M with SQL generates all fields, sub-files, and pointer files associated with the main file into relational tables.

### Creating The Link for All FileMan Files Within a Range

This option lets you link all FileMan files within a specified range to Open M with SQL. The range of files may be as small as one file or as large as your entire application.

To define the range, you specify the file numbers of the first and last files to be included. The files you specify must be main files; they cannot be sub-files.

**Note:** *To use this facility, you must know the file numbers of the files you want to link. It is not sufficient to know just the file names. To determine the file numbers, go to the FileMan Interface main menu and select the Print FileMan-Developer Cross Ref. option. This prints out a list of all FileMan files with their file numbers.*

**Procedure** To create the FileMan-Open M with SQL link for all FileMan files within a range:

1. Select the Link All FileMan Files to Developer option from the FileMan Interface menu.



You see the Link All Files to Developer window, as shown below:

|                                       |                           |
|---------------------------------------|---------------------------|
| +---Link All Files To Developer-----+ |                           |
| Starting File #                       | _____                     |
| Ending File #                         | _____                     |
| Compile Base Table(s)?                | No__                      |
| OK To Proceed?                        | No__                      |
| +-----+                               |                           |
| Main                                  | Press <PF1><PF3> For Help |

2. In the Starting File # field, type the file number of the starting file in the range of files you want to link to Open M with SQL.
3. In the Ending File # field, type the file number of the last file in the range of fields you want to link to Open M with SQL.

This must be a main file; it cannot be a sub-file.

This must be a main file; it cannot be a sub-file.

4. Answer *Yes* at the Compile Base Table(s)? prompt to compile all tables generated for the FileMan link.

This field defaults to *No*. Do not change this value if you want to leave the generated tables uncompiled.

5. Answer *Yes* at the O.K. To Proceed? prompt to initiate creation of the link.

You will see messages similar to the following as Open M with SQL builds the link:

```
CREATING CHILD TABLE FOR SUB-FILE #1.001
CHILD TABLE 'SUBDSRPT3' SUCCESSFULLY CREATED

CREATING CHILD TABLE FOR SUB-FILE #1.005
CHILD TABLE 'SB_PPLCTN_G' SUCCESSFULLY CREATED
BASE TABLE 'FILE' SUCCESSFULLY CREATED FOR FILE #1

CREATING BASE TABLE(S) FOR FILE #1.4

CREATING CHILD TABLE FOR SUB-FILE #1.41

CREATING CHILD TABLE FOR SUB-FILE #1.42
CHILD TABLE 'SB_SRS_LLWD' SUCCESSFULLY CREATED

CREATING CHILD TABLE FOR SUB-FILE #1.43
CHILD TABLE 'SB_SRS_DSSL' SUCCESSFULLY CREATED

CREATING CHILD TABLE FOR SUB-FILE #1.44
CHILD TABLE 'SB_GRP_S_LLW' SUCCESSFULLY CREATED

CREATING CHILD TABLE FOR SUB-FILE #1.45
CHILD TABLE 'SB_GRP_S_DSL' SUCCESSFULLY CREATED
CHILD TABLE 'SUB_NTRY_NM' SUCCESSFULLY CREATED
```

The linking messages are hierarchically indented to reflect levels of dependency in the created tables.

Messages that represent the creation of a parent table are pushed flush left.

Messages that represent the creation of a child table are indented one tab space.

Messages that represent the creation of a grandchild table are indented two tab spaces.

### Creating The Link One File at a Time

As a more conservative approach, you may create the FileMan-Open M with SQL link by specifying one file at a time.

This approach is most useful for extending the link, i.e., adding a new file after the original link is already in place.

To learn how to do this, see “Extending, Updating, and Deleting the FileMan-Open M Link” on page 19-13.

## Extending, Updating, and Deleting the FileMan-Open M Link

Open M with SQL provides an extensive facility for maintaining the FileMan-Open M with SQL link once it has been created.

This Link Maintenance facility lets you perform the following operations on the link:

- n You can **extend** the link to include new main files.

**Note:** You can also use this option to create the link on a file-by-file basis.

- n You can **update** the link by main file, sub-file, or field.
- n You can **delete** the link by main file, sub-file, or field.

### Link Maintenance Facility Runs Many Operations in Succession

Within the Link Maintenance facility, one operation (INSERT, UPDATE, DELETE) acts on a scope no larger than one main file. However, you may program multiple operations of the same or different kinds to occur in succession.

For example, you can add one or more main files, update fields within one or more main files, and delete sub-files of one or more main files all at the same time.

### Accessing the Link Maintenance Facility

To access the Link Maintenance facility, select the FileMan-Open M Developer Link option from the FileMan Interface menu.

The File Man Interface window appears, as shown below:

-----File Man Interface-----

FileName

Number

Update  
Indices?

Field

INSERT ( \_ )  
UPDATE ( \_ )  
DELETE ( \_ )

Field

( \_ )  
( \_ )  
( \_ )

Field

( \_ )  
( \_ )  
( \_ )

File Man Interface

Press <PF1><PF3> For Help

The following table lists and describes the fields located on the File Man Interface window:

Table 19-4: Fields Located on File Man Interface Window

| Field Name | Description                                                                                                                                                                                                                                                                                                                |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FileName   | Enter the name of the FileMan main file that you want to add, update or delete. You may press the <List Choices> key to see a lookup box that lists all main files.                                                                                                                                                        |
| Number     | Optionally, you may specify a main file by its file number instead of by its file name. Open M with SQL automatically fills in the file number if you specify the file by its name.                                                                                                                                        |
| INSERT     | Use the INSERT action to add an unlinked main file to the link. An INSERT action can only be performed when the specified file is not already linked. When selectable, INSERT appears in boldface type. Enter a value of X to enable this action. Leave blank if no action is desired.                                     |
| UPDATE     | Use the UPDATE action to update the link for an entire main file or a sub-file/field of that main file. An UPDATE action can only be performed when the specified file is already linked. When selectable, UPDATE appears in boldface type. Enter a value of X to enable this action. Leave blank if no action is desired. |
| DELETE     | Use the DELETE action to delete the link for a main file or a sub-file/field of that main file. A DELETE action can only be performed when the specified file is already linked. When selectable, DELETE appears in boldface type. Enter a value of X to enable this action. Leave blank if no action is desired.          |

**Table 19-4: Fields Located on File Man Interface Window**

| Field Name      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Field           | This field applies only when the main file specified in the FileName field has already been linked to Open M with SQL. Enter the name of the field or sub-file of the specified main file that you want to add, update, or delete from the link. Leave this field blank to specify all fields and sub-files associated with the main file. Use arrow syntax (->) when referring to fields in sub-files. Press the <List Choices> key to see a list of all possible entries. To specify that lower level sub-files should also be targeted, end the string with '...'. |
| Update Indices? | This is a Yes/No field. It applies only when you perform the UPDATE action on a file, sub-file, or field. Answer <i>Yes</i> to delete the old index maps and recreate them with the new definitions, and also to delete and recreate base table UPDATE triggers. Answer <i>No</i> to leave index maps unchanged.                                                                                                                                                                                                                                                      |

## Extending the FileMan-Open M with SQL Link

When you have created a FileMan-Open M with SQL link, you can extend the link by adding additional main files to it.

You can extend the link on a main file basis only, one file at a time.

When you add a main file to the link, Open M with SQL generates all fields, sub-files, and pointer files associated with the main file into relational tables.

### You Can Create the Link on a File-by-File Basis

You can also use this option to create the link from scratch. This is a conservative approach to creating the link, where you build the link on a file-by-file basis by individually selecting each file that you want to connect to Open M with SQL.

### **Procedure** To create/extend the FileMan-Open M with SQL link:

1. Select the FileMan-Open M Developer Link option from the FileMan Interface menu.

The FileMan Interface window appears, as shown below:

| -----File Man Interface----- |               |       |                 |
|------------------------------|---------------|-------|-----------------|
| FileName                     | Number        |       | Update Indices? |
| Inventory Record_____        | 54.2          |       |                 |
| Field _____                  | <b>INSERT</b> | ( X ) |                 |
|                              | UPDATE        | ( - ) | _____           |
|                              | DELETE        | ( - ) |                 |
| Patient_____                 | 2             |       |                 |
| Field _____                  | <b>INSERT</b> | ( x ) |                 |
|                              | UPDATE        | ( - ) | _____           |
|                              | DELETE        | ( - ) |                 |
| _____                        |               | ( - ) |                 |
| Field _____                  |               | ( - ) | _____           |
|                              |               | ( - ) |                 |

|                           |                                              |
|---------------------------|----------------------------------------------|
| <b>File Man Interface</b> | <b>Press &lt;PF1&gt;&lt;PF3&gt; For Help</b> |
|---------------------------|----------------------------------------------|

- At the FileName field, enter the name of the main file you want to link to Open M with SQL.

This must be a main file; it cannot be a sub-file.

You may press the <List Choices> key to see a lookup box that lists all FileMan main files.

Alternatively, you may select the main file by its file number.

- Type X at the INSERT field to specify the insert action.

INSERT appears in boldface type to indicate that it is an allowable action for the specified file. If it does not appear in boldface type, this means the file has already been linked, in which case UPDATE and DELETE are the only allowable actions.

- Press <RETURN> to specify the next main file.

You may specify as many main files for INSERT as you like, one file at a time.

- When you have specified all the main files that you want to link, press the <PROCEED> key.

You see the Compile window, as shown below:

| -----File Man Interface-----                                                            |                                                           |                    |
|-----------------------------------------------------------------------------------------|-----------------------------------------------------------|--------------------|
| FileName<br>Inventory Record_____                                                       | Number<br>54.2                                            | Update<br>Indices? |
| Field _____                                                                             | <b>INSERT</b> ( X )<br>UPDATE    ( - )<br>DELETE    ( - ) | _____              |
| Patient_____                                                                            | 2                                                         |                    |
| Field _____                                                                             | <b>INSERT</b> ( X )<br>UPDATE    ( - )<br>DELETE    ( - ) | _____              |
| +-----Compile-----+                                                                     |                                                           |                    |
| Compile Tables? _____                                                                   |                                                           |                    |
| O.K. to Proceed? _____                                                                  |                                                           |                    |
| +-----                                                                                  |                                                           |                    |
| <b>Compile</b> <span style="float: right;">Press &lt;PF1&gt;&lt;PF3&gt; For Help</span> |                                                           |                    |

6. Answer *Yes* at the Compile Tables? prompt to force Open M with SQL to compile all tables created for the FileMan link.

This field defaults to *No*. Do not change this value if you want to leave the generated tables uncompiled.

7. Answer *Yes* at the O.K. To Proceed? prompt to initiate creation of the link.

You see messages similar to the following as Open M with SQL builds the link:

```

inserting fileman File # INVENTORY RECORD
CREATING BASE TABLE(S) FOR FILE #54.2

    CREATING CHILD TABLE FOR SUB-FILE #54.21
    CHILD TABLE 'SUB_VAULT' SUCCESSFULLY CREATED

    CREATING CHILD TABLE FOR SUB-FILE #54.22
    CHILD TABLE 'SUB_QCK_CDS' SUCCESSFULLY CREATED
BASE TABLE 'INVNTY_RCR' SUCCESSFULLY CREATED FOR FILE #54.2

CONVERTING POINTER FIELDS TO DESIGNATIVE REFERENCES
DESIGNATIVE REFERENCES CONVERTED
  
```

The linking messages are hierarchically indented to reflect levels of dependency in the created tables.

Messages that represent the creation of a parent table are pushed flush left.

Messages that represent the creation of a child table are indented one tab space.

Messages that represent the creation of a grandchild table are indented two tab spaces.

### Updating the FileMan-Open M with SQL Link

Once you have linked one or more FileMan files to Open M with SQL, you can update the link at any time. This lets you keep the link current with any changes you make to the file definition structure at the FileMan level.

#### You Can Update the Link for a Main File, Sub-File, or Field

You can update the FileMan-Open M with SQL link at the level of any of the following objects:

- n An entire main file
- n A sub-file of the main file
- n A field within the main file or any of its sub-file(s)

When you update the link for an entire main file, Open M with SQL deletes the old link and recreates a new one.

When you update the link for a sub-file, Open M with SQL deletes and recreates the link for the starting sub-file as well as for each dependent sub-file recursively.

When you update the link for a field, Open M with SQL deletes and recreates just that field.

Using the Update Link action, you can do the following:

- n Update sub-files and/or fields that have been modified since the link was last updated.
- n Add new sub-files and/or fields to the link.

#### Selecting a Sub-file

To select a sub-file, use arrow syntax, as follows:

```
main_file->sub_file->...->field
```

#### **Procedure** To update the FileMan-Open M with SQL link:

1. Select the FileMan-M with SQL Link option from the FileMan Interface menu.



The FileMan Interface window appears, as shown below:

| -----File Man Interface-----       |                     |  |                 |
|------------------------------------|---------------------|--|-----------------|
| FileName                           | Number              |  | Update Indices? |
| Patient_____                       | 2                   |  |                 |
| Field _____                        | INSERT ( _ )        |  |                 |
|                                    | <b>UPDATE</b> ( X ) |  | Yes_            |
|                                    | DELETE ( _ )        |  |                 |
| Inventory Record_____              | 54.2                |  |                 |
| Field VAULT->QUANTITY ON HAND_____ | INSERT ( _ )        |  |                 |
|                                    | <b>UPDATE</b> ( X ) |  | Yes_            |
|                                    | DELETE ( _ )        |  |                 |
| Field _____                        | ( _ )               |  |                 |
|                                    | ( _ )               |  |                 |
|                                    | ( _ )               |  |                 |

**File Man Interface**
**Press <PF1><PF3> For Help**

2. At the FileName prompt, enter the name of the main file that you want to relink to Open M with SQL.

This must be a main file; it cannot be a sub-file.

You may press the <List Choices> key to see a lookup box that lists all FileMan main files.

Alternatively, you may select the main file by its file number.

3. At the Field prompt, enter the name of the field or sub-file within the specified main file that you want to link to Open M with SQL.

To relink the entire main file, leave this field blank.

To relink a selected sub-file (and its dependent sub-files recursively) or just a selected field, specify the sub-file or field here.

You may press the <List Choices> key to see a lookup box that lists all fields and sub-files associated with the specified main file.

4. Type X at the UPDATE field to specify the update action.

UPDATE appears in boldface type to indicate that it is an allowable action for the specified file. If it does not appear in boldface type, this means the file has not yet been linked, in which case INSERT is the only allowable action.

5. Answer *Yes* at the Update Indices? field to force Open M with SQL to delete and recreate the index maps and UPDATE triggers generated for the tables.

Leave this field blank if you do not want to update the index maps and UPDATE triggers.

6. Press <RETURN> to specify the next main file.

To relink a second field or sub-file from the same main file, reenter the name of the main file here, then specify another field or sub-file, as described in steps 2-6.

Or, you may specify a new main file, repeating steps 2-6.

7. When you have specified all the main files that you want to update, press the <PROCEED> key.

The Compile window appears.

8. Answer *Yes* at the Compile Tables? prompt to force Open M with SQL to recompile all updated tables.

By default, Open M with SQL does not compile these tables. You may leave this field blank to leave the tables uncompiled.

9. Answer *Yes* at the O.K. To Proceed? prompt to initiate update of the link.

Open M with SQL displays linking messages as it updates the link according to your specifications.

### Deleting the FileMan-Open M with SQL Link

Once you have created a FileMan-Open M with SQL link, you can delete it at any time.

#### **You Can Delete the Link for a Main File, Sub-File, or Field**

You can delete the FileMan-Open M with SQL link at the level of any of the following objects:

- n An entire main file
- n A sub-file of the main file
- n A field within the main file or any of its sub-file(s)

When you delete the link for an entire main file, Open M with SQL deletes the link for the main file and all its sub-files recursively.

When you deletes the link for a sub-file, Open M with SQL deletes the link for the starting sub-file as well as for each dependent sub-file recursively.

When you delete the link for a field, Open M with SQL deletes the link just for that field.

### Selecting a Sub-file

To select a sub-file, use arrow syntax, as follows:

```
main_file->sub_file->...->field
```

### Open M with SQL Enforces Referential Integrity When Deleting a File

When you delete the link at the level of a FileMan main file or sub-file, Open M with SQL deletes every field in the starting file, then deletes every dependent sub-file recursively. This recursive deletion prevents the existence of a child table whose parent has been deleted.

#### **Procedure** To delete the FileMan-Open M with SQL link:

1. Select the FileMan-M with SQL Link option from the FileMan Interface menu.

The FileMan Interface window appears, as shown below:

| -----File Man Interface----- |        |       |                 |
|------------------------------|--------|-------|-----------------|
| FileName                     | Number |       | Update Indices? |
| Patient_____                 | 2      |       |                 |
| Field _____                  | INSERT | ( - ) |                 |
|                              | UPDATE | ( - ) |                 |
|                              | DELETE | ( X ) |                 |
| Inventory Record_____        | 54.2   |       |                 |
| Field INVENTORY ITEM_____    | INSERT | ( - ) |                 |
|                              | UPDATE | ( - ) |                 |
|                              | DELETE | ( X ) |                 |
| _____                        |        | ( - ) |                 |
| Field _____                  |        | ( - ) |                 |
|                              |        | ( - ) |                 |

File Man Interface
Press <PF1><PF3> For Help

2. At the FileName field, enter the name of the main file whose link you want to delete.

This must be a main file; it cannot be a sub-file. You may press the <List Choices> key to see a lookup box that lists all FileMan main files.

Alternatively, you may select the main file by its file number.

3. In the field entitled Field, enter the name of the field or sub-file within the specified main file that you want to delete from the link.

To delete the entire main file, leave this field blank.

To delete a selected sub-file (and its dependent sub-files recursively) or just a selected field, specify the sub-file or field here.

You may press the <List Choices> key to see a lookup box that lists all fields and sub-files associated with the specified main file.

4. Type *X* at the DELETE field to specify the delete action.

DELETE appears in boldface type to indicate that it is an allowable action for the specified file. If it does not appear in boldface type, this means the file has not yet been linked, in which case INSERT is the only allowable action.

5. Press <RETURN> to complete the delete specification (the actual deletion is not performed until step 8).

To delete a second field or sub-file from the same main file, reenter the name of the main file here, then specify another field or sub-file, as described in steps 2-5.

Or, you may specify a new main file, repeating steps 2-5.

6. When you have specified all the main files that you want to delete, press the <PROCEED> key.

The Compile window appears.

7. Answer *Yes* at the Compile Tables? prompt to force Open M with SQL to recompile all updated tables.

By default, Open M with SQL does not compile these tables. You may leave this field blank to leave the tables uncompiled.

8. Answer *Yes* at the O.K. To Proceed? prompt to initiate deletion of the link.

Open M with SQL displays delete messages as it deletes the link according to your specifications.

## Generating Views of FileMan Files

The FileMan Interface provides a facility that automatically generates an Open M with SQL view of a FileMan file. The view joins all Open M with SQL tables created for the file. In Open M with SQL, you can use the view as the data source for reports, queries, and other views.

In order to generate a view of a file, the file must already be linked to Open M with SQL.

The view generation facility generates one view for every FileMan file you specify. You can specify a range of multiple files, in which case it will generate multiple views.

To define the range, you specify the file numbers of the first and last files to be included. These files must be main files; they cannot be sub-files.

**Note:** *To use this facility, you must know the file numbers of the files for which you want to generate views. It is not sufficient to know just the file names. To determine the file numbers, go to the FileMan Interface main menu and select the Print FileMan-Developer Cross Ref. option. This prints out a list of all FileMan files with their file numbers.*

### **Procedure** To generate views for FileMan files:

1. Select the Generate Views of FileMan Files option from the FileMan Interface menu.

You will see the FileMan View Generation window, as shown below:

```

+-----FileMan View Generation-----+
|                                     |
| Starting File # _____          |
| Ending File #   _____          |
| OK To Proceed?  Yes_               |
|                                     |
+-----+
FileMan View Creation                      Press <PF1><PF3> For Help

```

2. In the Starting File # field, type the file number of the first in the range of files for which you want to generate a view.

This must be a main file; it cannot be a sub-file.

3. In the Ending File # field, type the file number of the last in the range of files for which you want to generate a view.

This must be a main file; it cannot be a sub-file.

4. Answer *Yes* at the O.K. To Proceed? prompt to initiate generation of the view.

You see the following message as Open M with SQL generates the view:

```
Generating view...Please wait.
```

Then, you see:

```
View generation completed!
```

## FileMan-Open M with SQL Cross Reference Report

The FileMan-Open M with SQL Cross Reference Report shows the correspondence between FileMan files (both main files and sub-files) and the interface-generated Open M with SQL base tables equivalent to them.

For each FileMan main file and sub-file that has been linked to Open M with SQL, the FileMan-Open M with SQL Cross Reference Report prints the following information:

- n The file number
- n The file name
- n The name of the equivalent interface-generated Open M with SQL base table or child table

### **Procedure** To print the FileMan-Open M with SQL Cross Reference Report

1. Select the Print FileMan-Developer Cross Ref. option from the FileMan Interface menu.

You see the Device Selection window, as shown below:

|                                                                                                                                                                                                                                                                                                                                     |  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| <pre> +-----Device Selection-----+   Device                              /dev/tty9_____                                                     Description                         BJB-Feb 19                                                            Print Format                        _____                      +-----+ </pre> |  |
| <b>Device Selection</b> <span style="float: right;">Press &lt;PF1&gt;&lt;PF3&gt; For Help</span>                                                                                                                                                                                                                                    |  |
| Exit Without selecting                                                                                                                                                                                                                                                                                                              |  |

2. In the Device field, enter the name of the device to which you want to send this report.

You can send the report to any valid output device to which your current device is linked.

The default device is always your current device. To print the report to your screen, press the <PROCEED> key without altering the device name.

The FileMan-Developer Cross Reference Report appears as follows:

| FileMan - Open M Developer Cross Reference Report               |                           |                       | Page: 1 |
|-----------------------------------------------------------------|---------------------------|-----------------------|---------|
| Printed: 02/19/97 02:39PM                                       |                           |                       |         |
| FileMan Number                                                  | File Manager File Name    | Base Table Name       |         |
| .2                                                              | DESTINATION               | DESTINATION_2         |         |
| .4                                                              | PRINT TEMPLATE            | PRINT_TEMPLATE_4      |         |
| .401                                                            | SORT TEMPLATE             | SORT_TEMPLATE_401     |         |
| .402                                                            | INPUT TEMPLATE            | INPUT_TEMPLATE_402    |         |
| .41                                                             | FORMATTER-FIELD SUB-FIELD | SUB_FORMATTERFIELD_41 |         |
| .5                                                              | FUNCTION                  | FUNCTION_5            |         |
| .7                                                              | MUMPS OPERATING SYSTEM    | MUMPS_OPERATING_SYSTE |         |
| .8                                                              | SCREEN                    | SCREEN_8              |         |
| .81                                                             | DISPLAY SUB-FIELD         | SUB_DISPLAY_81        |         |
| .87                                                             | WINDOW SUB-FIELD          | SUB_WINDOW_87         |         |
| .871                                                            | '@'-NAMES SUB-FIELD       | SUB_NAMES_871         |         |
| .88                                                             | COMMAND ACTION            | COMMAND_ACTION_88     |         |
| .881                                                            | COMMAND SCREEN SUB-FIELD  | SUB_COMMAND_SCREEN_88 |         |
| .882                                                            | SYNONYMS SUB-FIELD        | SUB_SYNONYMS_882      |         |
| Press <Return> to continue, <Options> to scroll, <Exit> to Exit |                           |                       |         |



# Conversion Error Log

## Print Conversion Error Log Report

Use this option to print a report of all errors encountered during INSERT, UPDATE, and DELETE operations performed on the FileMan-Open M with SQL link.

The Conversion Error Log Report provides the following information about each error:

- n Date
- n Time
- n Directory
- n FileMan File Number
- n Error Type
- n Error Description
- n SQL Error Code

### **Procedure** To print the Conversion Error Log Report

1. Select the Print Conversion Error Log Report option from the FileMan Interface menu.

You see the Device Selection window, as shown below:

|                                                                                                                   |  |
|-------------------------------------------------------------------------------------------------------------------|--|
| <pre> +-----Device Selection-----+ Device /dev/tty9_____ Description BJB-Feb 19 Print Format _____ +-----+ </pre> |  |
| <div>Device Selection</div> <div>Press &lt;PF1&gt;&lt;PF3&gt; For Help</div>                                      |  |
| Exit Without selecting                                                                                            |  |

2. In the Device field, enter the name of the device to which you want to send this report.

You can send the report to any valid output device to which your current device is linked.

The default device is always your current device. To print the report to your screen, press the <PROCEED> key without altering the device name.

The Print Conversion Error Log Report appears as follows:

```
FileManager Interface Errorpage: 1
Printed: 02/19/97 02:46PM
Date: 01/10/97 Time: 08:03PM Directory: /dbms1/cust/saic/
File #: 44.2 (PATIENT APPOINTMENT)
Error Type: Message
updating fileman File # PATIENT APPOINTMENT - Updating indices
SQLCODE = 0
Date: 01/10/97 Time: 08:03PM Directory: /dbms1/cust/saic/
File #: 44.2 (PATIENT APPOINTMENT)
Error Type: Warning Only
Invalid FileMan Index. Omitting Index #2 On Field #.01.
SQLCODE = 0
Date: 01/10/97 Time: 08:03PM Directory: /dbms1/cust/saic/
File #: 44.2 (PATIENT APPOINTMENT)
Error Type: Fatal Error
Press <Return> to continue, <Options> to scroll, <Exit> to Exit
```

### Purge Conversion Error Log

Use this option to purge the conversion error log of all entries prior to a specified date. The system requests confirmation before purging the error log.

#### **Procedure** To purge the conversion error log:

1. Select the Purge Conversion Error Log option from the FileMan Interface menu.

The Purge FileMan Error Log window appears, as shown below:

|                                     |                           |
|-------------------------------------|---------------------------|
| +-----Purge FileMan Error Log-----+ |                           |
| Purge Errors Up To Date: _____      |                           |
| Ok To Proceed? No__                 |                           |
| +-----+                             |                           |
| Main                                | Press <PF1><PF3> For Help |

2. Enter the date up to which you want to purge all errors from the log at the Purge Errors Up To Date prompt.  
  
To enter the current date, type *t*. This purges all errors from the log.
3. Answer Yes at the OK to Proceed? prompt.
4. Press <RETURN> on the <proceed> prompt located on the bottom boundary of the window to start the purge.

When the purge is complete, you see the following message at the bottom of the screen:

```
Error Log Purged.
```

## Using the FileMan Interface with Open M with SQL PDP

Using the FileMan Interface with Open M with SQL PDP has special requirements. Hardware resource limitations prevent you from defining and compiling base tables or forms on the Digital PDP-11 platform.

The workaround for this is to define and compile base tables and forms on another development machine, then export them to the PDP-11 production machine. The export utility downloads both globals and macro source routines. You must compile the macro source routines on the PDP-11 machine to generate PDP-executable object code. Once you have done this, you can run compiled forms and define new reports and queries from compiled base tables on the PDP-11 production machine.

Since the FileMan Interface generates and compiles base tables, you must run the interface on a non-PDP-11 development machine. You can then export the base tables and macro routines to the PDP-11 machine and compile the macro routines on the PDP-11. Please consult the chapter entitled “Export/Import” in the *Open M/SQL Developer Guide* for information on using the export utility.

Since FileMan routines and globals are accessed both during creation of the interface and during execution of reports and forms, you need to copy the complete set of FileMan routines and globals onto both the development machine and the production machine.

## Limitations to the FileMan Interface

The following is a list of limitations associated with the FileMan Interface:

- n The FileMan pointer data type is supported. However, additional logic for screened pointers and restricted "Learn-As-You-Go" (LAYGO) access is not supported. Screened pointers and LAYGO access are controlled at the form level.
- n Computed fields are supported. However, only the following standard FileMan variables are automatically available for computations:
  - U
  - DT
  - D0 - Dn
- n Special lookup programs are not supported.
- n Post-selection programs are not supported.
- n Neither file level security access nor field level security access is supported. Open M with SQL's security system is compliant with the ANSI SQL standard, and supersedes FileMan security.
- n Output transformations are supported. However, the only FileMan variables which are automatically available are X and Y. Any other required variables must be established in the application environment prior to invoking Open M with SQL.
- n The only cross-reference index structure supported for Open M with SQL lookups and retrievals is the regular index. The following index types are not supported for lookups and retrievals, but are maintained for database integrity:
  - Kwik
  - Mnemonic
  - Soundex
  - Trigger
  - Bulletin Board
- n FileMan currently permits a programmer to define standard delimited fields and fixed-length fields in the same positions of a global node, allowing one field to overwrite another. Open M with SQL does not rectify such database design errors.
- n Any M programs referenced within FileMan field validation or transformation logic, computed field logic, or index maintenance logic must be accessible from the Open M with SQL directory where the interface-generated base tables reside.

- n Under certain circumstances FileMan allows the same global node to be used for multiple cross-reference indices. This confuses Open M with SQL when using a cross-reference index to perform a lookup.
- n Under certain circumstances FileMan allows a cross-reference index for a file to be maintained through one of its sub-files. Because such an index is not relationally correct, it is not used by Open M with SQL for lookups or searches on the sub-file. However, the index is maintained to enforce data-base integrity.
- n Field auditing is not supported.
- n Basic help messages are supported, but executable help messages are not supported.
- n Word processing fields are supported only as text fields. No genuine word processing capability is currently available within Open M with SQL.

## FileMan Interface Questions and Answers

**Q:** Can I use Open M with SQL for generating reports and queries, while continuing to use my FileMan application for data entry?

**A:** Yes, the interface does not diminish the viability of existing FileMan applications. You may also use the Open M with SQL Form Generator to create new data entry applications for your database.

**Q:** Once I run the interface against a FileMan database, can I modify the tables within Open M with SQL? What happens to my FileMan definitions if I do this?

**A:** You can modify any base table generated by the FileMan Interface. However, the FileMan-Open M with SQL link is a one-way link, and there is no interface to send changes from Open M with SQL back to FileMan. InterSystems recommends making any changes first to the FileMan definition, then updating the FileMan-Open M with SQL link appropriately.

**Q:** If there are problems with my FileMan definitions, can I revise them and re-run the interface?

**A:** Yes, you can use the UPDATE option to relink a file or part of a file at any time.

**Q:** What are the limitations to using the FileMan Interface under Open M with SQL PDP on the Digital PDP-11 platform?

**A:** Due to hardware limitations, you must create the link on a non-PDP-11 development machine, then use the Export/Import utility to copy the generated base tables and macro source routines to a production environment on the PDP-11 machine. You must have a copy of all FileMan routines and globals on both the development and production machines.

**Q:** Why doesn't Open M with SQL support screened pointers and restricted LAYGO access?

**A:** Open M with SQL supports both of these features. However, Open M with SQL differs from FileMan in how it approaches these features. FileMan makes these features an integral part of the database, while Open M with SQL considers these as application-dependent features. Therefore, Open M with SQL allows these features to be defined at the form level, rather than at the database level.

**Q:** Why doesn't Open M with SQL support special lookup programs and post-selection programs?

A: Again, Open M with SQL takes the approach that these are application-dependent, and not intrinsic to the database definition.

Q: Why doesn't Open M with SQL support FileMan's security features?

A: Open M with SQL employs its own database security system, which is compliant with the ANSI SQL standard. FileMan's security scheme is not compatible with this standard.

Q: Why doesn't Open M with SQL support non-regular index types for searches and retrievals?

A: In order to convert a FileMan index to an index map, Open M with SQL must be able to parse a global reference from the executable M code used by FileMan to maintain indices. The nature of the non-regular indices makes it impractical to obtain a global reference for them. Instead, Open M with SQL calls the FileMan index maintenance code through base table Post-Filing triggers.

Q: Why doesn't Open M with SQL support FileMan's field auditing feature?

A: Currently, Open M with SQL makes field auditing the responsibility of the application developer.

Q: Why doesn't Open M with SQL support FileMan's executable help messages?

A: Open M with SQL help messages are lines of constant text. No provision currently is made for executing M code when displaying help.





---

## Importing Data Definitions

---

Often, when users want to convert existing M applications to the Open M with SQL RDBMS, they want to transfer existing data definitions. You can automate this transfer to Data Dictionary base tables by writing a program that creates a global, named ^mxdd, which contains the information in the format expected by the Data Dictionary. This chapter discusses the format of this global. Topics covered include:

|                                          |            |
|------------------------------------------|------------|
| n Overview of Data Dictionary Import     | page 20-2  |
| n Basic Structure of ^mxdd Import Global | page 20-2  |
| n Base Table Node Structure              | page 20-5  |
| n View Node Structure                    | page 20-25 |
| n Privileges Definition Node Structure   | page 20-30 |

## Overview of Data Dictionary Import

Users often want to transfer their existing data definitions into the Data Dictionary so they can use the many features available with InterSystems' Open M Developer environment.

### Create ^mxdd Global

In order to do this, users must write an application to place the information they currently have defined into the global, ^mxdd. Users can then run the Open M with SQL import utility, which will take the information in the ^mxdd global to create a Data Dictionary definition.

## Basic Structure of ^mxdd Import Global

The ^mxdd global describes two types of objects:

- n Base tables
- n Views

### Provide Definition as Described in Data Dictionary

The values you provide to describe base tables and views must be in the form described in this guide for the corresponding window and field.

For instance, when you provide the format of a field of data type Date, you will enter a number from 1 - 7, to represent one of the seven formats described under the Date data type in “Defining a Date Field” on page 6-11 in Chapter 6, *Defining Base Table Fields*.

### Values Provided as Subscripts and Node Values

Object information may be in the form of node value and/or subscript value.

For instance, when you describe a base table, you will provide the base table name as the second subscript value, as in:

```
^mxdd(1,<Base Table Name>,...)
```

You may reference a field name. Enclose the field name in curly braces, as in {fieldname}. If the field is in another base table, use the table.fieldname format, as in {Other\_table.fieldname}.

## Abbreviations Used to Represent Values

In the descriptions of the ^mxdd global in this document, the following subscript values refer to definition information you provide:

```
<ObjectType>="View" or "BaseTable"
<Name>= Object Name (either name of View or name of Base
Table)
<BName>  = Base Table Name
<VName>  = View Name
<FName>  = Field Name
<MName>  = Map Name
<VFName> = View Field Name
<Node>   = Node Name
```

Open M with SQL runs each base table or view name through the \$\$ALPHAUP function. The resulting name must not match any of the SQL reserved words. See Appendix C, *SQL Reserved Words*.

## Fixed Subscript Values

Many nodes have fixed subscript values, which will be enclosed in quotation marks ("). These are case sensitive. For example, the subscript "ViewFields" is different from the subscript "viewfields". Be sure to enter subscripts in quotes exactly as they appear in this document.

## Identify Object as Base Table or View

You define whether a node refers to a base table or a view, it will be the value of the node:

```
^mxdd(1,<Name>)=<ObjectType>
```

where <ObjectType> can equal "View" or "BaseTable". Note the first subscript of ^mxdd is currently fixed with the value of 1.

Subsequent subscripts define different types of information about the base table or view named in the second subscript. For instance, each base table field is described in subnodes of the following node:

```
^mxdd(1,<BName>,"Field",<FName>,<Node>)
```

## Required and Default Values

The descriptions of each node of the ^mxdd global indicate when information is required and must be included as a node in ^mxdd. If you do not include these nodes, you will receive an error message when you attempt to import the ^mxdd global.

Sometimes a node is marked **REQUIRED** but the information is only required based on the value given to another node.

Some nodes contain information that is required when entered in a Data Dictionary window, but for which Open M with SQL provides a default value. Therefore, you do not have to provide a value for these fields. These nodes will not be marked as required in the descriptions that follow, but their default values are provided.

### Multi-Line Node Structure

Most nodes contain actual values. However, some nodes represent a multi-line field, then the value of the node will be a number representing the number of lines in the field. There then are that number of lower-level nodes, with subscripts starting at 1, containing the value of each of the lines. For example, if # is a node in ^mxdd which represents a multi-line field, then:

```
^mxdd(....,#) = n, where n is the number of lines in this
multi-line field
^mxdd(....,#,1) = value of first line
^mxdd(....,#,2) = value of second line
.
.
.
^mxdd(....,#,n) = value of last line
```

## Base Table Node Structure

The definition of a base table consists of nodes which describe:

- n Basic definition
- n Field definitions
- n Field error message translations
- n Field help message translations
- n Trigger definitions
- n Mapping definition
  - Basic definition
  - Access Path Specifications
  - Data Access Variables
  - Row ID Specifications
  - Map Data Specifications
  - Index Maps

### Base Table Basic Definition

The basic definition describes the main features of the base table. The top-level node structure of a Base Table definition is:

```
^mxdd( 1 , <BName> , # )
```

#### Location in Data Dictionary

In the Data Dictionary, you define most of these values at the Base Table Definition master window. The fileman fields you define in the Fileman Interface window, accessed via the Fileman-M with SQL Link option on the Fileman Interface menu. The Fileman Interface menu is an option on the Data Dictionary menu.

### Node Descriptions

The meaning of the various nodes represented by different values of # is provided in the table "Base Table Top-Level Nodes" below:

**Table 20-1: Base Table Top-Level Nodes**

| #  | How to Access from the Base Table Definition Master Window                                   | Field Name on Window                  | Notes                                                 |
|----|----------------------------------------------------------------------------------------------|---------------------------------------|-------------------------------------------------------|
| 3  | On the window                                                                                | Description                           |                                                       |
| 11 | Validation Code horizontal menu option displays "Base Table Validation Code" window          | "Insert" Validation M code Lines      | (MULTILINE)                                           |
| 13 | Implicit Joins horizontal menu option displays "Characteristic Joins" window                 |                                       | Name of the Parent Table (if it exists)               |
| 14 | Implicit Joins horizontal menu option displays "Characteristic Joins" window                 | Names of Child Tables                 | (MULTILINE)                                           |
| 31 | Validation Code horizontal menu option displays "Base Table Validation Code" window          | "Update" Validation M code Lines      | (MULTILINE)                                           |
| 32 | On the window                                                                                | Approximate number of rows            |                                                       |
| 34 | Validation Code horizontal menu option displays "Base Table Validation Code" window          | "Delete" Validation M code Lines      | (MULTILINE)                                           |
| 35 | Do not access from Base Table Definition Master Window, but at the File-man Interface window | Number                                | Number of File-man file                               |
| 41 | Advanced Options horizontal menu option displays "Advanced Base Table Options" window        | Routine Name                          | Routine name prefix                                   |
| 44 | On the window                                                                                | Use default physical structure? (Y/N) | REQUIRED                                              |
| 47 | <Physical Structure> branching field                                                         | File Name                             | Default file name (Meaningful only if node 44 = "Y"). |
| 48 | Advanced Options horizontal menu option displays "Advanced Base Table Options" window        | BaseTable Owner                       | Owner of this table                                   |

**Example** This node provides the name, description, parent table, approximate number of rows, routine name prefix, whether using default structure, and base table owner.

```
^mxdd(1,"MachCompts",3)=Connector table to Parts
13)=Machines
32)=500
41)=mt15
44)=N
48)=_SYSTEM
```

## Base Table Field Definitions

The field definition node structure is identified by the "Field" value in the third subscript.

```
^mxdd(1,<BName>,"Field",<FName>,<Node>)
```

### Data Dictionary Location

In the Data Dictionary, you define most of these values at the Field Definition master window. The fileman fields you define in the Fileman Interface window, accessed via the Fileman-M with SQL Link option on the Fileman Interface menu. The Fileman Interface menu is an option on the Data Dictionary menu.

### Node Descriptions

The meaning of the various nodes represented by different values of # is provided in the table "Base Table Field Nodes":

**Table 20-2: Base Table Field Nodes**

| # | How to Access from Field Definition Master Window                        | Field Name on Window | Notes                                                                                                                                                                                                                                                       |
|---|--------------------------------------------------------------------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3 | On the window                                                            | Description          |                                                                                                                                                                                                                                                             |
| 4 | Update Features horizontal menu option displays "Update Features" window | Column Number        | <p>REQUIRED (Output only on Field Definition form)</p> <p>Input Value Meaning</p> <p><b>0</b> Designative reference to parent table</p> <p><b>1</b> Row ID</p> <p><b>2</b> childsub child subscript field</p> <p><b>Unique integer</b> All other fields</p> |

Table 20-2: Base Table Field Nodes

| #  | How to Access from Field Definition Master Window                                               | Field Name on Window        | Notes                                                                                                                                     |
|----|-------------------------------------------------------------------------------------------------|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| 6  | On the window                                                                                   | Data type                   | REQUIRED<br><b>Input Values</b><br>Date<br>Designative Reference<br>Multiple Choice<br>Name<br>Number<br>Row ID<br>Text<br>Time<br>Yes/No |
| 7  | On the window                                                                                   | Maximum length              | REQUIRED                                                                                                                                  |
| 8  | Update Features horizontal menu option displays "Update Features" window horizontal menu option | Field protection            | <b>Input Value    Protection Level</b><br>AE            Add/Edit<br>A             Add only<br>O             Output only                   |
| 9  | Update Features horizontal menu option displays "Update Features" window                        | Required?                   | <b>Input Value    Meaning</b><br>Y            Yes<br>N            No<br>M            Maybe                                                |
| 10 | Update Features horizontal menu option displays "Update Features" window                        | M code for "Maybe" required | REQUIRED if field is Maybe Required.                                                                                                      |
| 11 | On the window                                                                                   | Multi-line field?           | (Y/N)                                                                                                                                     |
| 14 | Multi-Line Storage window                                                                       | Storage type                | REQUIRED if it is a multi-line field<br><b>Input Value    Meaning</b><br>1            All in One Node<br>2            One per Node        |
| 15 | Multi-Line Storage window                                                                       | Storage delimiter           | REQUIRED if multi-line field with "All in One Node" storage type                                                                          |



Table 20-2: Base Table Field Nodes

| #  | How to Access from Field Definition Master Window | Field Name on Window                                                    | Notes                                                                                       |
|----|---------------------------------------------------|-------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| 19 | Time data type window                             | First time                                                              | Format <i>HH:MMAM</i> or <i>HH:MMPM</i> For a "Time" data type                              |
| 20 |                                                   | Last time                                                               | Format <i>HH:MMAM</i> or <i>HH:MMPM</i> For a "Time" data type                              |
| 21 |                                                   | Format                                                                  | For a "Time" data type<br><b>Input Value</b><br>A digit from the range 1-6                  |
| 22 | Number data type window                           | Number minimum                                                          |                                                                                             |
| 23 |                                                   | Number maximum                                                          |                                                                                             |
| 24 |                                                   | # of decimal places                                                     |                                                                                             |
| 25 |                                                   | Format                                                                  |                                                                                             |
| 26 |                                                   | Leading punctuation                                                     |                                                                                             |
| 27 | Number data type window                           | Show minus                                                              |                                                                                             |
| 28 |                                                   | Used in arithmetic calculation?                                         | (Y/N)<br>For a "Number" data type                                                           |
| 29 | Multiple Choice data type window                  | Multiple choices                                                        | (MULTILINE)<br>For "Multiple Choice" data type                                              |
| 30 | Date data type window                             | First Date                                                              | Format <i>MM/DD/YYYY</i><br>For "Date" data type                                            |
| 31 |                                                   | Date last                                                               |                                                                                             |
| 33 |                                                   | Format                                                                  | For "Date" data type                                                                        |
| 36 | Designative Reference data type window            | Designated base table name                                              | REQUIRED<br>For "Designative Reference" data type                                           |
| 38 | On the window                                     | Approximate Number of distinct values                                   |                                                                                             |
| 43 | On the window                                     | Computed field?                                                         | (Y/N)                                                                                       |
| 45 | Computed Field window                             | Names of the fields whose change triggers the computation of this field | (MULTILINE)<br>For a computed field of type "Not Always Equal"-- see Node 82 in this table) |

Table 20-2: Base Table Field Nodes

| #  | How to Access from Field Definition Master Window                                              | Field Name on Window                      | Notes                                                                                                          |
|----|------------------------------------------------------------------------------------------------|-------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| 46 | Computed Field window                                                                          | Code to compute the value of the field    | (MULTILINE) (for a computed field)                                                                             |
| 49 | Help/Error Messages horizontal menu option displays "Error and Help Messages" window           | Error message (English)                   |                                                                                                                |
| 50 |                                                                                                | Help message (English)                    | (MULTILINE)                                                                                                    |
| 51 | Conversion/Validation Code horizontal menu option displays "Conversion/Validation Code" window | External to Internal Conversion M code    | (MULTILINE)                                                                                                    |
| 52 | Conversion/Validation Code horizontal menu option displays "Conversion/Validation Code" window | Internal to External Conversion M code    | (MULTILINE)                                                                                                    |
| 61 | On the window                                                                                  | Unique?                                   | (Y/N)                                                                                                          |
| 62 | Row ID data type window                                                                        | Field displayable?                        | (Y/N)<br>For Row ID data type                                                                                  |
| 63 | Multiple Choice data type window                                                               | Do you want external and internal values? | (Y/N)<br>For Multiple Choice data type                                                                         |
| 72 | Report Defaults horizontal menu option                                                         | Column title                              | For an M/PACT report                                                                                           |
| 73 | Row ID data type window                                                                        | Is it based on other fields?              | (Y/N)<br>REQUIRED if value in 6 is Row ID.                                                                     |
| 74 | Row ID data type window                                                                        | Fields                                    | Names of fields on which Row ID is based.<br>REQUIRED if value in 73 is Yes and in 6 is Row ID.<br>(MULTILINE) |
| 75 |                                                                                                | Auxiliary data type                       | For Row ID data type<br>For input values, see #6 above                                                         |

Table 20-2: Base Table Field Nodes

| #  | How to Access from Field Definition Master Window                                                                 | Field Name on Window                     | Notes                                                                                                         |
|----|-------------------------------------------------------------------------------------------------------------------|------------------------------------------|---------------------------------------------------------------------------------------------------------------|
| 79 | Text and Name data type windows                                                                                   | Function used for collating and look-ups | <b>Input Meaning</b><br>A ALPHAUP<br>E EXACT (default)<br>U UPPER                                             |
| 80 | Conversion/Validation Code horizontal menu option displays "Conversion/Validation Code" window                    | Additional Validation code               | (MULTILINE)                                                                                                   |
| 81 | Update Features horizontal menu option displays "Update Features" window                                          | M expression for default value           | (External)<br>If Row ID, defaults to:<br>\$\$next("^mdata(<TableID>)")<br>where <TableID> is the table's irn. |
| 82 | Computed Field window                                                                                             | Is value always equal to computation?    | (Y/N)<br>REQUIRED for computed fields                                                                         |
| 83 |                                                                                                                   | "Compute as null IF" M expression        | (MULTILINE)<br>For computed field only                                                                        |
| 86 | Do not access from Base Table Definition Master Window, but at the Fileman Interface window                       | Number                                   | FileMan field number                                                                                          |
| 88 | 1. Error/Help Messages horizontal menu option displays "Error and Help Messages" window                           | Help Override routine                    |                                                                                                               |
| 89 | 2. <Edit Override Help>                                                                                           | Array to store override help             |                                                                                                               |
| 96 | Do not access from Base Table Definition Master Window, but from Fileman Interface option on Data Dictionary menu | INSERT                                   | FileMan general triggers (MULTILINE)                                                                          |
| 97 |                                                                                                                   | UPDATE<br>DELETE                         | FileMan update triggers (MULTILINE)                                                                           |

**Example** These nodes describe the field "MachCompnts", which is the Row ID field for the base table "MachCompnts".

```
^mxd(1,"MachCompnts","Field","MachCompnts",
3)=MachCompnts Row ID
4)=1
6)=Row ID
7)=16
8)=0
9)=Y
11)=N
25)=1
27)=1
28)=Y
38)=NUMROWS
43)=N
49)=Enter a number such as 1234.56
61)=Y
73)=Y
74)=2
74,1)=Machines
74,2)=childsub
```

**Example** These nodes describe the Machines field, which is a designative reference field to the parent table. (Note that node 4, column number, is therefore given the value "0".)

```

^mxd(1,"MachCompnts","Field","Machines",
3)=Machines Parent Reference
4)=0
6)=Designative Reference
7)=16
8)=0
9)=Y
11)=N
28)=Y
36)=Machines
43)=N
49)=Press <CHOICE> for a list of choices.
51)=6
51,1)=n c,k,neg,p,reg
2)=s reg=0 s:%val=+%val reg=1 q:reg
3)=s neg=1 s:%val<0 neg=-1,%val=$e(%val,2,999)
4)=s:%val'?n.l'.'.n %ok=0
5)=s:%val?.p %val="" s:%val&$g(neg)%val=%val*neg
6)=s %val=+%val
62)=N
75)=Number
81)=$g(da12)

```

## Base Table Field Error Translations

The field error translation definition node structure is identified by the "FErrTrans" value in the fifth subscript. The translation, like the English version of the error message, may be one line long.

### Data Dictionary Location

In the Data Dictionary, you define Error Message translations at the window displayed when you press <Error Translations> on the Error and Help Messages window. See "Defining Long Help and Error Messages" on page 13-4 in Chapter 13, *Field Help and Error Messages*

### <Language> Subscript

Replace the <Language> subscript with the name of the language in which this error message is written.

### Node Descriptions

The value of this node is one line which is the error message that will be displayed if invalid data is entered into a field.

```
^mxd(1,<BName>,"Field",<FName>,"FErrTrans",  
      <Language>)=error message
```

**Example** This node shows an error message translated into German.

```
^mxd(1,"Machines","Field","Notes","FErrTrans","Deutsch  
(German)")  
      =German error message
```

## Base Table Field Help Translations

The field help translation definition node structure is a multi-line structure, identified by the "FHelpTrans" value in the fifth subscript.

### Data Dictionary Location

In the Data Dictionary, you define the Help translations at the window displayed when you press <Long Help Translations> on the Error and Help Messages window. See “Defining Long Help and Error Messages” on page 13-4 in Chapter 13, *Field Help and Error Messages*.

### <Language> Subscript

Replace the <Language> subscript with the name of the language in which this error message is written.

### Node Descriptions

This is a multi-line node. The first node contains a counter of the number of lines in the help message. The subnodes contain the lines of the help message.

```
^mxd(1,<BName>,"Field",<FName>,"FHelpTrans",  
      <Language>)=number of lines in help message  
^mxd(1,<BName>,"Field",<FName>,"FHelpTrans",  
      <Language>,n)=help message line
```

**Example** This German help message contains 2 lines:

```
^mxdd(1,"Machines","Field","Notes","FHelpTrans","Deutsch
(German)")=2
^mxdd(1,"Machines","Field","Notes","FHelpTrans","Deutsch
(German)",
1)=German help message - first line
2)=German help message - second line
```

## Base Table Triggers Definition

The trigger definition node structure is identified by a <TriggerType> value (see the section “<TriggerType> Subscript” on page 20-15) in the third subscript.

```
^mxdd(1,<BName>,<TriggerType>,<TriggerItem#>,#)
```

### Data Dictionary Location

In the Data Dictionary, you define triggers at the window displayed when you select the Base Table Triggers horizontal menu option at the Base Table Definition master window. See Chapter 11, *Base Table Triggers* .

### <TriggerType> Subscript

The <TriggerType> subscript can have the values described in the table "<TriggerType> Subscript Values":

**Table 20-3: <TriggerType> Subscript Values**

| Value      | Description              |
|------------|--------------------------|
| PreInsert  | Pre-file CREATE trigger  |
| PreUpdate  | Pre-file UPDATE trigger  |
| PreDelete  | Pre-file DELETE trigger  |
| PostInsert | Post-file CREATE trigger |
| PostUpdate | Post-file UPDATE trigger |
| PostDelete | Post-file DELETE trigger |

### <TriggerItem#> Subscript

You should number trigger items sequentially, starting from 1 for each new value of <TriggerType>.

Node Descriptions

The meaning of the various nodes represented by different values of # is provided in Table 20-4: Base Table Triggers Nodes, on page 20-16.

These fields are found on the Trigger Definition window. To get to this window select the Base Table Triggers horizontal menu option from Base Table Definition master window. Press <RETURN> at any of the six branching fields described in the table "<TriggerType> Subscript Values>. The Trigger Definition window appears:

-----Base Table Definition-----

Base Table Name

Description

Accountsaccounts base table

Use Default Physical Structure? No

< Fields >

-----Trigger Definition-----

Action

Form/Window/Field

TypeQuery/Routine(M Expressions)

IF

IF

IF

IF

IF

ImplicitValidationBaseTableCompileComp RelCopyAdvancedComments

...Trigger DefinitionPress <PF1><PF3> For Help

Table 20-4: Base Table Triggers Nodes

| # | Field Name on "Trigger Definition" Window | Description                                               |
|---|-------------------------------------------|-----------------------------------------------------------|
| 2 | Access Code ("IF ...)                     |                                                           |
| 3 | Action Type                               | REQUIRED. Input values:<br>n Routine<br>n M Code<br>n SQL |
| 4 | Routine name                              | REQUIRED if Action Type is "Routine".                     |
| 7 | M Code Lines                              | REQUIRED if Action Type is "M Code".<br>(MULTILINE)       |
| 8 | SQL Lines                                 | REQUIRED if Action Type is "SQL".<br>(MULTILINE)          |



**Example** This node describes a PreDelete trigger. It is of type SQL, and is one line long.

```
^mxdd(1, "Machines", "PreDelete", 1,
3)=SQL
8)=1
8,1)=DELETE FROM InstldSftwr WHERE Machine=:%data(1)
```

## Base Table Map Node Structure

Include this node only if you are defining customized mapping, i.e., the value of the node ^mxdd(1,<BName>,44)="N".

The map node structure definition is identified by the literal value "Map" in the third subscript.

```
^mxdd(1, <BName>, "Map", <MName>, #)
```

### Data Dictionary Location

In the Data Dictionary, you define maps at the Map Definition row selection window, which you access by selecting <Physical Structure> at the Base Table Definition master window.

### Node Descriptions

The meaning of the various nodes represented by different values of # is provided in the table "Base Table Map Nodes":

**Table 20-5: Base Table Map Nodes**

| # | How to Access from Map Definition Master Window | Field Name on Window  | Notes                                                                        |
|---|-------------------------------------------------|-----------------------|------------------------------------------------------------------------------|
| 4 | On window                                       | Global name           | REQUIRED                                                                     |
| 6 | On window                                       | Is this a Master Map? | (Y/N)<br>REQUIRED<br>There must be exactly 1 Master Map for each base table. |

**Table 20-5: Base Table Map Nodes**

| #  | How to Access from Map Definition Master Window | Field Name on Window            | Notes                                                                                                                                           |
|----|-------------------------------------------------|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| 15 | 1. Advanced Options on horizontal menu          | Conditional Map?                | (Y/N)<br>Only if value of Node 6 is "N".                                                                                                        |
| 22 | 2. <Conditional Map>                            | Set Map IF                      | M expression to evaluate the condition (if Conditional Map node = "Y")                                                                          |
| 25 |                                                 | Percentage of rows in this map. | Provide the percentage with a leading decimal point. For example ".25" will translate to 25%.                                                   |
| 28 | Advanced Options on horizontal menu             | INSERT                          | See "Step 10: Define Override Filing Code" in Chapter 18, <i>Creating a Customized Map Definition</i> .                                         |
| 29 |                                                 | UPDATE                          |                                                                                                                                                 |
| 30 |                                                 | DELETE                          |                                                                                                                                                 |
| 32 | <Full Row Reference>                            | Override                        | Row reference override value. See "Step 7: Specify Override to Full Row Reference" in Chapter 18, <i>Creating a Customized Map Definition</i> . |

**Example** This node describes the Master Map named "Machines Master Map". The data is in the global ^mxddMachine, and it is not a conditional map.

```

^mxdd(1,"Machines","Map","Machines Master Map",
4)=^mxddMachine
6)=Y
15)=N

```

## Base Table Map Access Path Specifications Structure

The map Access Path Specifications structure is identified by a "Map" value in the third subscript and a "MapSubs" value in the fifth subscript.

```

^mxdd(1,<BName>,"Map",<MName>,"MapSubs",<AccessLevel>,#)

```

## Data Dictionary Location

In the Data Dictionary, you define Access Levels at the Access Path Specifications window, which you access from the Map Definition master window. See “Access Path Specifications” on page 16-17 in Chapter 16, *Default Physical Structure*.

### <AccessLevel> Value

This is the number of the Access Level. You start with the number 1, and continue until you have defined all Access Levels necessary to provide the Full Row Reference.

## Node Descriptions

The meaning of the various nodes represented by different values of # is provided in the table "Base Table Map Access Path Specification Nodes":

**Table 20-6: Base Table Map Access Path Specification Nodes**

| #  | How to Access from Access Path Specifications Master Window   | Field Name on Window                      | Notes                                                                                |
|----|---------------------------------------------------------------|-------------------------------------------|--------------------------------------------------------------------------------------|
| 2  | On window                                                     | Level                                     | REQUIRED<br>(Output Only on window)                                                  |
| 8  | On window                                                     | Value Expression                          | REQUIRED<br>You can reference a field, as in {fieldname} or {Other_table.fieldname}. |
| 9  | <Access Code>                                                 | First Value                               |                                                                                      |
| 10 |                                                               | Last Value                                |                                                                                      |
| 16 | 1. <Access Code><br>2. <Special Code>                         | Special invalid condition code - override | (MULTILINE)                                                                          |
| 18 | 1. <Access Code><br>2. <Special Code><br>3. <NEXT Subroutine> | Next subroutine - override value          | (MULTILINE)                                                                          |

**Table 20-6: Base Table Map Access Path Specification Nodes**

| #  | How to Access from Access Path Specifications Master Window | Field Name on Window | Notes                                                  |
|----|-------------------------------------------------------------|----------------------|--------------------------------------------------------|
| 20 | Piece auxiliary window                                      | Piece delimiter      | REQUIRED for Access Type "PIECE" -- see node 23 below) |
| 22 | <Data Access Expression>                                    | Override Reference   | Override value for Data Access Expression              |
| 23 | On window                                                   | Access Type          | Input Values<br>Global<br>Sub<br>Piece<br>Other        |

**Example** These nodes describe three master map subscripts for the base table "MachCompnts".

```

^mxdd(1,"MachCompnts","Map","Machine Comp Master
Map","MapSubs",1,
2)=1
8)=0
^mxdd(1,"MachCompnts","Map","Machine Comp Master
Map","MapSubs",2,
2)=2
8)={Machines.Machines}
^mxdd(1,"MachCompnts","Map","Machine Comp Master
Map","MapSubs",3,
2)=3
8)={childsub}

```

These nodes describe 2 subscripts for the index map of the base table "Machines" named "Index InHseName 15".

```
^mxdd(1,"Machines","Map","Index InHseName
15","MapSubs",1,
2)=18)="Inhse machine name"
^mxdd(1,"Machines","Map","Index InHseName
15","MapSubs",2,
2)=2
8)=$$UPPER({InHseNa})
```

## Base Table Map Data Access Variable

The map data access variable structure is identified by the "MapVars" value in the seventh subscript. It is a multi-line node.

```
^mxdd(1,<BName>,"Map",<MName>,"MapSubs",
<AccessLevel>,"MapVars",<VariableNumber>)= count
^mxdd(1,<BName>,"Map",<MName>,"MapSubs",
<AccessLevel>,"MapVars",<VariableNumber>,n)= M Code
```

### Data Dictionary Location

To access the Additional Data Access Variables window in the Data Dictionary, you:

1. Enter the Base Table Definition master window
2. Select <Physical Structure>
3. If the table has default physical structure, select <Examine Structure>
4. Specify the Map Name
5. Select <Access Path Specifications>
6. Select <Access Code>
7. Select <Additional Data Variables - M Code>

### <VariableNumber> Value

This is the number of a Data Access Variable for the given <AccessLevel> value. Assign 1 to the first variable, 2 to the second, and so on.

**Example** These nodes describe 1 Data Access Variable. Its value is one line long.

```
^mxdd(1,"Machines","Map","Machines Master
Map","MapSubs",
      2,"MapVars",1)=1
^mxdd(1,"Machines","Map","Machines Master
Map","MapSubs",
      2,"MapVars",1,1)=SET {2D1}={S2}_a"
```

Base Table Row ID Specifications

The Row ID specifications definition is identified by the "MapRowID" value in the fifth subscript.

```
^mxdd(1,<BName>,"Map",<MName>,"MapRowID",
      <MRowIDSub>,#)
```

Data Dictionary Location

The values for these fields are defined at the Row ID Specifications window, accessed from the Map Definition master window.

Node Descriptions

The meaning of the various nodes represented by different values of # is provided in the table "Base Table Row ID Specification Nodes". These fields are both on the Row ID Specifications window.

Table 20-7: Base Table Row ID Specification Nodes

| # | Field Name on "Row ID Specifications" Window | Notes    |
|---|----------------------------------------------|----------|
| 3 | Field                                        | REQUIRED |
| 4 | M expression                                 | REQUIRED |

**Example** This node describes a Row ID field named Machines which is not based on other fields. It is equal to the Value Expression defined for Access Level 3 of the Access Path Specifications.

```
^mxdd(1,"Machines","Map","Index InHseName
15","MapRowID",1,3)=Machines
      4)={L3}
```

## Base Table Map Data Specifications Structure

The Map Data Specifications structure is identified by the "MapData" value in the fifth subscript.

```
^mxd(1, <BName>, "Map", <MName>, "MapData",
    <MapDataSubscript>, #)
```

### Data Dictionary Location

These values are all defined at the Map Data Specifications window, accessed from the Map Definition master window.

### Node Descriptions

The meaning of the various nodes represented by different values of # is provided in the table "Base Table Map Data Nodes":

**Table 20-8: Base Table Map Data Nodes**

| #  | How to Access from Map Data Specifications Master Window | Field Name on Window | Notes                                    |
|----|----------------------------------------------------------|----------------------|------------------------------------------|
| 2  | On the window                                            | Field name           | REQUIRED                                 |
| 3  | On the window                                            | Node                 |                                          |
| 4  | On the window                                            | Piece                |                                          |
| 5  | On the window                                            | Delimiter            | REQUIRED if a value is given for node 4. |
| 10 | <M Code> branching field                                 | M Code               | (MULTILINE)                              |

**Example** In this example, node 5 (delimiter definition) is required since there is a value given to node 4.

```
^mxd(1, "Machines", "Map", "IndexInHseName15", "MapData", 1,
    2)=InHseName
    4)=1
    5)="^mxd"
```

## Default Physical Structure Index Definitions

This node is required only if you plan to use default physical structure. Include this if you gave a value of "Y" to the node ^mxd(1,<BName>,44).

The index definition for a base table created with default physical structure is identified by a "MapDefIdxFd" value in the third subscript. It is a multi-line node.

```
^mxd(1, <BName>, "MapDefIdxFd", <IndexNumber>) = count  
^mxd(1, <BName>, "MapDefIdxFd", <IndexNumber>, #) = Field Name
```

### Data Dictionary Location

When you are using default physical structure, you create additional index maps in the Data Dictionary by following these steps:

1. Select <Physical Structure> at the Base Table Definition master window
2. Select <Fields to Index>
3. Enter one or more groups of fields. Each group creates a separate Index Map.

### <IndexNumber> Value

This is the number of the Index Map, starting from 1.

**Example** These nodes describe one index map.

```
^mxd(1, "MachCompnts", "MapDefIdxFd", 1) = 1  
^mxd(1, "MachCompnts", "MapDefIdxFd", 1, 1) = Machines
```



## View Node Structure

There are three types of views:

- n Views based on a base table
- n Views based on another view
- n Views based on a query (General Views)

### View Basic Definition

The basic definition exists of all three types of views. It describes the main features of the view. The top-level node structure of a View definition is:

```
^mxd(1,<VName>,"ViewName",#)
```

### Data Dictionary Location

These values are defined at or from the View Definition master window.

### Node Descriptions

The meaning of the various nodes represented by different values of # is provided in the table "View Top-Level Nodes" below. These fields are all found on the View Definition form.

**Table 20-9: View Top-Level Nodes**

| # | How to Access from the View Definition Master Window | Field          | Notes                                                                                                      |
|---|------------------------------------------------------|----------------|------------------------------------------------------------------------------------------------------------|
| 5 | On the window                                        | Description    |                                                                                                            |
| 6 | On the window                                        | Starting table | REQUIRED<br>Base table name if view is based on a base table or View name if view is based on another view |
| 8 | <Query> branching field                              | Query text     | REQUIRED if a query-based view (MULTILINE)                                                                 |

**Table 20-9: View Top-Level Nodes**

| #  | How to Access from the View Definition Master Window | Field       | Notes                                                                                                                                                          |
|----|------------------------------------------------------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11 | On the window                                        | Start Type: | Output only on window.<br><b>Input Meaning</b><br>1 View is based on a base table<br>3 View is based on another view<br>8 View is a General (Query-based) View |
| 14 | Advanced Options horizontal menu option              | Owner       | Output only in Data Dictionary                                                                                                                                 |

**Example**      `^mxd(1,"MachManuf",5)=Machines with manufacturers shown  
                  6)=Machines  
                 11)=1  
                 14)=_SYSTEM`

## View Join Specification

You need to provide these nodes which define view joins only if the view is based on a base table, but not if the view is based on another view or a query.

The view join specification is identified by the "Link" value in the third subscript.

```
^mxd(1,<VName>,"Link",<LinkNumber>,#)
```

### Data Dictionary Location

You define these fields at the View Join Specification window, accessed by selecting <Join Specifications> at the View Definition master window.

### <LinkNumber> Definition

Replace <LinkNumber> with a unique, sequential integers, starting with 1, for each base table join comprising this view. You will use this number when you

describe the fields from each joined table (including the starting one) that are included in this view. See the section “View Field Definition” on page 20-27.

### Node Descriptions

The meaning of the various nodes represented by different values of # is provided in the table "View Join Specification Nodes" below:

**Table 20-10: View Join Specification Nodes**

| # | Field Name on "View Join Specifications" Window | Meaning                               |
|---|-------------------------------------------------|---------------------------------------|
| 2 | Base Table                                      | REQUIRED                              |
| 3 | Join Description                                |                                       |
| 5 | Implicit Join                                   | REQUIRED<br>Use extended arrow syntax |

**Example** These nodes describe a designative reference to the Machines base table.

```
^mxdd(1, "MachManuf", "Link", 1, 2) = Vendors
    3) = to get manufacturer's name
    5) = Machines.VendorID
```

These nodes describe a link using extended arrow syntax.

```
^mxdd(1, "MachManuf", "Link", 1, 2) = Parts
    3) = parts table
    5) = Machines.MachCompnts->PartID
```

## View Field Definition

The definition of view fields exists only if the view is based on a base table or another view, but not if the view is based on a query.

View fields descend hierarchically in global nodes for views based on a base table. All fields belong to link #1 for views based on another view.

The view field definition is identified by the "ViewFields" value in the fifth subscript.

```
^mxdd(1, <VName>, "Link", <LinkNumber>, "ViewFields",
    <VFName>, #)
```

### Data Dictionary Location

You enter the values in these nodes at the Fields on View Definition window. You reach that window by pressing <Fields> at the View Join Specification window.

### <LinkNumber> Definition

#### Views Based on Base Table

Replace <LinkNumber> with the number of each base table you defined in your View Links Definition. See the section “View Join Specification” on page 20-26.

#### Views Based on a View

Replace <LinkNumber> with the number 1. You will define all fields included in this view under this number.

### Node Descriptions

The meaning of the various nodes represented by different values of # is provided in the table "View Field Nodes":

**Table 20-11: View Field Nodes**

| # | Field Name on "Fields on View Definition" Window | Meaning                                                                                                                                                                                                                                            |
|---|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3 | Description                                      |                                                                                                                                                                                                                                                    |
| 4 | Field Name in Linked Table                       | REQUIRED<br>The name of the underlying base table field on which this view field is based, in the form: <BName>_". "_<FName><br>For views based on another view, <BName> refers to the underlying base tables in view on which this view is based. |
| 7 | Not on window                                    | REQUIRED if view is based on another view only <LinkNumber> of base table on which underlying view is based from which the field is derived                                                                                                        |
| 8 | Field Name in View                               | REQUIRED<br>Name of the view field from the view on which this view is based on which this field is based (for view-based views only).                                                                                                             |

**Example** This is a view based on a base table. Therefore, there are no nodes 7 and 8 to define this field ("BrandName"), but only a node 4 (Node 3 -- field description -- is always optional and was not included).

```
^m added(1, "MachManuf", "Link", 1, "ViewFields", "BrandName", 4) =  
Machines.BrandName
```

# Privileges Definition Node Structure

Privileges may be defined for a specific base table or view, as follows:

```
^mxdd(1,<BName or VName>,"Privilege")=<Number of
privileges>
^mxdd(1,<BName or VName>,"Privilege",<PrivilegeNumber>,#)
```

## Data Dictionary Location

Define all values, except Grantor, at the GRANT window, accessed via the Grant Privileges option on the Privileges Menu.

## Node Descriptions

The meaning of the various nodes represented by different values of # is provided in the table "Privilege Nodes" below:

Table 20-12: Privilege Nodes

| #           | Field Name on "GRANT" Window | Notes                                                                                                                                                                                                                                                                                                |             |         |     |        |     |        |     |        |     |        |     |        |     |            |
|-------------|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|---------|-----|--------|-----|--------|-----|--------|-----|--------|-----|--------|-----|------------|
| 3           | GRANT (Action)               | REQUIRED<br><table><tr><th>Input Value</th><th>Meaning</th></tr><tr><td>"a"</td><td>%ALTER</td></tr><tr><td>"s"</td><td>SELECT</td></tr><tr><td>"i"</td><td>INSERT</td></tr><tr><td>"u"</td><td>UPDATE</td></tr><tr><td>"d"</td><td>DELETE</td></tr><tr><td>"r"</td><td>REFERENCES</td></tr></table> | Input Value | Meaning | "a" | %ALTER | "s" | SELECT | "i" | INSERT | "u" | UPDATE | "d" | DELETE | "r" | REFERENCES |
| Input Value | Meaning                      |                                                                                                                                                                                                                                                                                                      |             |         |     |        |     |        |     |        |     |        |     |        |     |            |
| "a"         | %ALTER                       |                                                                                                                                                                                                                                                                                                      |             |         |     |        |     |        |     |        |     |        |     |        |     |            |
| "s"         | SELECT                       |                                                                                                                                                                                                                                                                                                      |             |         |     |        |     |        |     |        |     |        |     |        |     |            |
| "i"         | INSERT                       |                                                                                                                                                                                                                                                                                                      |             |         |     |        |     |        |     |        |     |        |     |        |     |            |
| "u"         | UPDATE                       |                                                                                                                                                                                                                                                                                                      |             |         |     |        |     |        |     |        |     |        |     |        |     |            |
| "d"         | DELETE                       |                                                                                                                                                                                                                                                                                                      |             |         |     |        |     |        |     |        |     |        |     |        |     |            |
| "r"         | REFERENCES                   |                                                                                                                                                                                                                                                                                                      |             |         |     |        |     |        |     |        |     |        |     |        |     |            |
| 4           | TO (User)                    | REQUIRED                                                                                                                                                                                                                                                                                             |             |         |     |        |     |        |     |        |     |        |     |        |     |            |
| 5           | Grantor name                 | REQUIRED<br>Name of the grantor of this privilege                                                                                                                                                                                                                                                    |             |         |     |        |     |        |     |        |     |        |     |        |     |            |
| 6           | WITH GRANT OPTION?           | REQUIRED<br>(Y/N)                                                                                                                                                                                                                                                                                    |             |         |     |        |     |        |     |        |     |        |     |        |     |            |

**Example**

```
^mxdd(1,"MachCompnts","Privilege")=1
^mxdd(1,"MachCompnts","Privilege",1,3)=a
4)=sales
5)=_SYSTEM
6)=N
```

---

# Data Dictionary Specifications

---

## Data Dictionary Specifications

| System Parameters                           | Maximum Value                               |
|---------------------------------------------|---------------------------------------------|
| Number of Fields per Base Table             | 150                                         |
| Length of Base Table/View Name              | 40 characters                               |
| Length of Field Name                        | 40 characters                               |
| Number of options for Multiple Choice field | 1023                                        |
| Length of Multiple Choice field option      | 75 characters                               |
| Number of Fields to Lookup On               | 7                                           |
| Number of fields to display in Lookup Box   | Up to screen width.<br>No multi-line fields |





## Keyboard Actions

Open M/SQL supports many terminals. To see a complete list of supported terminals, do the following:

1. Enter the following command:

```
>DO ^%
```

2. Enter a ? at the "Terminal type vt100=>" prompt.

The tables in this appendix show the keyboard mapping of form actions for the terminal types supported by Open M/SQL.

- n The first column, "Form Action," lists the actions supported by the Form Generator.
- n The second column, "Primary Key," lists the primary keystroke or keystroke sequence used to invoke each action.
- n The third column, "Alternate Key(s)," lists the alternate keystroke(s) or keystroke sequence(s) used to invoke each action.

### Altos

#### Altos Keyboard Mapping

| Form Action                         | Primary Key          | Alternate Key(s)   |
|-------------------------------------|----------------------|--------------------|
| Beginning of Field                  | <Enhance>-Left Arrow | <Enhance>-<Ctrl-H> |
| Bottom of List                      | <Enhance>-SCRN Next  | <Enhance>-B        |
| Bottom of Window                    | <Enhance>-Down Arrow | <Enhance>-<Ctrl-J> |
| Delete Row                          | Del Line             | <Enhance>-D        |
| Delete Instance (Multi-line fields) | <Enhance>-Del Line   |                    |

## Appendix B—Keyboard Actions

---

| Form Action                         | Primary Key           | Alternate Key(s)   |
|-------------------------------------|-----------------------|--------------------|
| Delete Word                         | <Ctrl-W>              |                    |
| Delete Previous Character           | Del                   |                    |
| Delete Character                    | <Ctrl-D>              | Del Char           |
| Down Arrow                          | Down Arrow            | <Ctrl-J>           |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K> |
| Enhance                             | F13                   | <Ctrl-G>           |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Del      |
| Explain                             | F14                   |                    |
| GETOUT                              | F10                   | <Enhance>-F        |
| GETOUTALL                           | <Enhance>-F10         |                    |
| Go to Bottom Menu                   | F15                   |                    |
| Help Menu                           | <Enhance>-F15         |                    |
| Insert Instance (Multi-line fields) | <Enhance>-Ins Line    |                    |
| Insert Row (Multi-row forms)        | Ins Line              | <Enhance>-I        |
| Left Arrow                          | Left Arrow            | <Ctrl-H>           |
| List Choices                        | <Enhance>-L           |                    |
| Next Field                          | Tab                   |                    |
| Next Word                           | <Ctrl-F>              |                    |
| Next Screen                         | SCRN Next             | <Enhance>-N        |
| Previous Screen                     | SCRN Prev             | <Enhance>-P        |
| Previous Word                       | <Ctrl-B>              |                    |
| PREVIOUS                            | F16                   |                    |
| PROCEED                             | F12                   |                    |
| RETURN                              | Retn<br>Enter         |                    |
| Right Arrow                         | Right Arrow           | <Ctrl-K>           |
| SAVE                                | F11                   | <Enhance>-S        |
| Search Current Table                | <Enhance>-F14         |                    |
| Top of Window                       | <Enhance>-Up Arrow    | <Enhance>-<Ctrl-U> |
| Top of List                         | <Enhance>-SCRN Prev   | <Enhance>-T        |
| Typeover/Insert Mode Toggle         | <Enhance>-^           |                    |

---

| Form Action | Primary Key | Alternate Key(s) |
|-------------|-------------|------------------|
| Undo        | <Ctrl-X>    |                  |
| Up Arrow    | Up Arrow    | <Ctrl-U>         |

## ANSI

### ANSI Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)               |
|-------------------------------------|-----------------------|--------------------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  |                                |
| Bottom of List                      | <Enhance>-Page Down   | <Enhance>-V                    |
|                                     |                       | <Ctrl-V>                       |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J>             |
| Delete Row (Multi-row forms)        | F8                    |                                |
| Delete Instance (Multi-line fields) | <Enhance>-F8          |                                |
| Delete Word                         | <Ctrl-W>              | <Enhance>-W                    |
| Delete Previous Character           | <--                   |                                |
| Delete Character                    | Del Char              | <Ctrl-D><br><Enhance>-D        |
| Down Arrow                          | Down Arrow            | <Ctrl-J>                       |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K>             |
| Enhance                             | F1                    | <Ctrl-G>                       |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Del<br><Enhance>-<-- |
| Explain                             | F2                    |                                |
| GETOUT                              | F10                   |                                |
| GETOUTALL                           | <Enhance>-F10         |                                |
| Go to Bottom Menu                   | F3                    |                                |
| Help Menu                           | <Enhance>-F3          |                                |
| Insert Instance (Multi-line fields) | <Enhance>-Insert      |                                |
| Insert Row (Multi-row forms)        | Insert                | <Enhance>-I                    |
| Left Arrow                          | Left Arrow            |                                |

## Appendix B—Keyboard Actions

---

| Form Action                 | Primary Key        | Alternate Key(s)        |
|-----------------------------|--------------------|-------------------------|
| List Choices                | F7                 |                         |
| Next Field                  | Tab                |                         |
| Next Word                   | <Ctrl-F>           | <Enhance>-F             |
| Next Screen                 | Page Down          | <Enhance>-N             |
| Previous Screen             | Page Up            | <Enhance>-P             |
| Previous Word               | <Ctrl-B>           | <Enhance>-B             |
| PREVIOUS                    | F4                 |                         |
| PROCEED                     | F5                 |                         |
| RETURN                      | Return             |                         |
| Right Arrow                 | Right Arrow        | <Ctrl-K>                |
| SAVE                        | F9                 |                         |
| Search Current Table        | F6                 | <Enhance>-F2            |
| Top of Window               | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U>      |
| Top of List                 | Page Up            | <Enhance>-T<br><Ctrl-T> |
| Typeover/Insert Mode Toggle | <Ctrl-^>           |                         |
| Undo                        | <Ctrl-X>           | <Enhance>-X             |
| Up Arrow                    | Up Arrow           | <Ctrl-U>                |

### CIT-500

#### CIT-500 Keyboard Mapping

| Form Action               | Primary Key          | Alternate Key(s)   |
|---------------------------|----------------------|--------------------|
| Beginning of Field        | <Enhance>-Left Arrow | <Enhance>-<Ctrl-J> |
| Bottom of Window          | <Enhance>-Down Arrow | <Enhance>-<Ctrl-J> |
| Bottom of List            | <Enhance>-Center     | <Enhance>-B        |
| Break                     | <Ctrl-C>             |                    |
| Delete Character          | <Ctrl-D>             |                    |
| Delete Word               | <Ctrl-W>             |                    |
| Delete Previous Character | Backspace            |                    |

| Form Action                         | Primary Key           | Alternate Key(s)   |
|-------------------------------------|-----------------------|--------------------|
| Delete Instance (Multi-line fields) | <Enhance>-Under Score |                    |
| Delete Row (Multi-row forms)        | Under Score           |                    |
| Down Arrow                          | Down Arrow            | <Ctrl-J>           |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K> |
| Enhance                             | PF1                   |                    |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Del      |
| Explain                             | PF2                   |                    |
| GETOUT                              | F20                   | <Enhance>-F        |
| GETOUTALL                           | <Enhance>-F20         |                    |
| Go to Bottom Menu                   | PF3                   |                    |
| Help Menu                           | F15<br><Enhance>-F15  |                    |
| Insert Row (Multi-row forms)        | EOL                   | <Enhance>-I        |
| Insert Instance (Multi-line fields) | <Enhance>-EOL         |                    |
| Left Arrow                          | Left Arrow            | <Ctrl-H>           |
| List Choices                        | Home                  | <Enhance>-L        |
| Next Field                          | Tab                   |                    |
| Next Screen                         | Center                | <Enhance>-N        |
| Next Word                           | <Ctrl-F>              |                    |
| Previous Word                       | <Ctrl-B>              |                    |
| Previous Screen                     | EOP                   | <Enhance>-P        |
| PREVIOUS                            | PF4                   |                    |
| PROCEED                             | F16                   |                    |
| RETURN                              | Return                |                    |
| Right Arrow                         | Right Arrow           | <Ctrl-K>           |
| SAVE                                | F19                   | <Enhance>-S        |
| Search Current Table                | F9                    | <Enhance>-PF2      |
| Top of List                         | <Enhance>-EOP         | <Enhance>-T        |
| Top of Window                       | <Enhance>-Up Arrow    | <Enhance>-<Ctrl-U> |
| Typeover/Insert Mode Toggle         | <Enhance>-^           |                    |

| Form Action | Primary Key             | Alternate Key(s) |
|-------------|-------------------------|------------------|
| Undo        | <Ctrl-X><br><Enhance>-X |                  |
| Up Arrow    | Up Arrow                | <Ctrl-U>         |

## COBRA

### COBRA Keyboard Mapping

| Form Action                         | Primary Key                | Alternate Key(s)                        |
|-------------------------------------|----------------------------|-----------------------------------------|
| Beginning of Field                  | <Enhance>-Left Arrow       |                                         |
| Bottom of List                      | <Enhance>-PageDn           | <Enhance>-B                             |
| Bottom of Window                    | <Enhance>-Down Arrow       | <Enhance>-<Ctrl-J>                      |
| Break                               | <Ctrl-C><br>Break <Ctrl-C> |                                         |
| Delete Row (Multi-row forms)        | F8                         | <Enhance>-D                             |
| Delete Instance (Multi-line fields) | <Enhance>-F8               |                                         |
| Delete Word                         | <Ctrl-W>                   |                                         |
| Delete Character                    | <Ctrl-D>                   |                                         |
| Delete Previous Character           | Backspace                  | Delete                                  |
| Down Arrow                          | Down Arrow                 | <Ctrl-J>                                |
| End of Field                        | <Enhance>-Right Arrow      | <Enhance>-<Ctrl-K>                      |
| Enhance                             | F1                         | <Ctrl-G>                                |
| Erase Field Line                    | <Ctrl-L>                   | <Enhance>-Delete<br><Enhance>-Backspace |
| Explain                             | F2                         |                                         |
| GETOUT                              | F10                        | <Enhance>-F                             |
| GETOUTALL                           | F11                        |                                         |
| Go to Bottom Menu                   | F3                         |                                         |
| Help Menu                           | F12                        |                                         |
| Insert Instance (Multi-line fields) | <Enhance>-INS              |                                         |
| Insert Row (Multi-row forms)        | INS                        | <Enhance>-I                             |

---

| Form Action                 | Primary Key        | Alternate Key(s) |
|-----------------------------|--------------------|------------------|
| Left Arrow                  | Left Arrow         |                  |
| List Choices                | F7                 | <Enhance>-L      |
| Next Field                  | Tab                |                  |
| Next Word                   | <Ctrl-F>           |                  |
| Next Screen                 | PageDn             | <Enhance>-N      |
| Previous Screen             | PageUp             | <Enhance>-P      |
| Previous Word               | <Ctrl-B>           |                  |
| PREVIOUS                    | F4                 |                  |
| PROCEED                     | F5                 | Esc-Esc          |
| RETURN                      | Return             |                  |
| Right Arrow                 | Right Arrow        | <Ctrl-K>         |
| SAVE                        | F9                 | <Enhance>-S      |
| Search Current Table        | F6                 |                  |
| Top of Window               | <Enhance>-Up Arrow |                  |
| Top of List                 | <Enhance>-PageUp   | <Enhance>-T      |
| Typeover/Insert Mode Toggle | <Enhance>-^        |                  |
| Undo                        | <Enhance>-X        |                  |
| Up Arrow                    | Up Arrow           |                  |

## DECTERM

### DECTERM (DEC-PC7XL-AA) Keyboard Mapping

| Form Action               | Primary Key          | Alternate Key(s)      |
|---------------------------|----------------------|-----------------------|
| Beginning of Field        | <Enhance>-Left Arrow | <Enhance>-<Ctrl-H>    |
| Bottom of List            | <Enhance>-B          | <Enhance>-Next Screen |
| Bottom of Window          | <Enhance>-Down Arrow | <Enhance>-<Ctrl-J>    |
| Break                     | <Ctrl-C>             |                       |
| Delete Word               | <Ctrl-W>             |                       |
| Delete Character          | <Ctrl-D>             |                       |
| Delete Previous Character | Backspace            |                       |

## Appendix B—Keyboard Actions

---

| Form Action                         | Primary Key           | Alternate Key(s)   |
|-------------------------------------|-----------------------|--------------------|
| Delete Row (Multi-row forms)        | <Enhance>-D           |                    |
| Delete Instance (Multi-line fields) | <Enhance>-Delete      |                    |
| Down Arrow                          | Down Arrow            | <Ctrl-J>           |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K> |
| Enhance                             | F9                    |                    |
| Erase Field Line                    | <Ctrl-L>              |                    |
| Explain                             | F10                   |                    |
| GETOUT                              | <Enhance>-F           |                    |
| GETOUTALL                           | <Enhance>-F8          |                    |
| Go to Bottom Menu                   | <Enhance>-F9          |                    |
| Help Menu                           | <Enhance>-F11         |                    |
| Insert Row (Multi-row forms)        | <Enhance>-I           |                    |
| Insert Instance (Multi-line fields) | <Enhance>-Insert      |                    |
| Left Arrow                          | Left Arrow            | <Ctrl-H>           |
| List Choices                        | <Enhance>-L           |                    |
| Next Field                          | Tab                   |                    |
| Next Word                           | <Ctrl-F>              |                    |
| Next Screen                         | Next Screen           | <Enhance>-N        |
| Previous Screen                     | Prev Screen           |                    |
| Previous Word                       | <Ctrl-B>              |                    |
| PREVIOUS                            | <Enhance>-Tab         |                    |
| PROCEED                             | <Enhance>-P           |                    |
| RETURN                              | Return                |                    |
| Right Arrow                         | Right Arrow           | <Ctrl-K>           |
| SAVE                                | <Enhance>-S           |                    |
| Search Current Table                | <Enhance>-F10         |                    |
| Top of Window                       | <Enhance>-Up Arrow    | <Enhance>-<Ctrl-U> |
| Top of List                         | <Enhance>-T           |                    |
| Typeover/Insert Mode Toggle         | Insert                |                    |



---

| Form Action | Primary Key | Alternate Key(s) |
|-------------|-------------|------------------|
| Undo        | <Enhance>-X | <Ctrl-X>         |
| Up Arrow    | Up Arrow    | <Ctrl-U>         |

## DTM-PC Console

See NT Console.

## DESQView Console

See PC Console.

## ED3638

### ED3638 (Edisa) Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)                     |
|-------------------------------------|-----------------------|--------------------------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  | ^E Left Arrow                        |
| Bottom of List                      | <Enhance>-V           | <Ctrl-V><br><Enhance>-Page Down      |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J><br>^E Down Arrow  |
| Break                               | <Ctrl-C>              |                                      |
| Delete Row (Multi-row forms)        | <Enhance>-D           | F8                                   |
| Delete Instance (Multi-line fields) | <Enhance>-F8          |                                      |
| Delete Word                         | <Enhance>-W           | <Ctrl-W>                             |
| Delete Character                    | Delete                | <Ctrl-D>                             |
| Delete Previous Character           | <--                   |                                      |
| Down Arrow                          | Down Arrow            | <Ctrl-J>                             |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K><br>^E Right Arrow |
| Enhance                             | F1                    | <Ctrl-G>                             |
| Erase Field Line                    | <Enhance>-Delete      | <Ctrl-L><br><Enhance><--             |
| Explain                             | F2                    |                                      |
| GETOUT                              | F10                   |                                      |

## Appendix B—Keyboard Actions

---

| Form Action                         | Primary Key        | Alternate Key(s)                  |
|-------------------------------------|--------------------|-----------------------------------|
| GETOUTALL                           | <Enhance>-F10      |                                   |
| Go to Bottom Menu                   | <Enhance>-F1       |                                   |
| Help Menu                           | <Enhance>-F3       |                                   |
| Insert Instance (Multi-line fields) | <Enhance>-Insert   |                                   |
| Insert Row (Multi-row forms)        | Insert             | <Enhance>-I                       |
| Left Arrow                          | Left Arrow         |                                   |
| List Choices                        | F7                 |                                   |
| Next Field                          | Tab                |                                   |
| Next Word                           | <Enhance>-F        | <Ctrl-F>                          |
| Next Screen                         | Page Down          | <Enhance>-N                       |
| Previous Screen                     | Page Up            | <Enhance>-P                       |
| Previous Word                       | <Ctrl-B>           |                                   |
| PREVIOUS                            | F4                 | <Enhance>-Tab                     |
| PROCEED                             | F5                 |                                   |
| RETURN                              | Return             |                                   |
| Right Arrow                         | Right Arrow        | <Ctrl-K>                          |
| SAVE                                | F9                 |                                   |
| Search Current Table                | F6                 |                                   |
| Top of Window                       | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U><br>^E Up Arrow |
| Top of List                         | <Enhance>-T        | <Enhance>-Page Up<br><Ctrl-T>     |
| Typeover/Insert Mode Toggle         | <Ctrl-^>           |                                   |
| Undo                                | <Enhance>-X        | <Ctrl-X>                          |
| Up Arrow                            | Up Arrow           | <Ctrl-U>                          |

---

## FALCO

### Falco Keyboard Mapping (Emulates DEC VT 220)

| Form Action                         | Primary Key           | Alternate Key(s)   |
|-------------------------------------|-----------------------|--------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  | <Enhance>-<Ctrl-H> |
| Bottom of List                      | <Enhance>-Next Screen | <Enhance>-B        |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J> |
| Break                               | <Ctrl-C>              |                    |
| Delete Word                         | <Ctrl-W>              |                    |
| Delete Instance (Multi-line fields) | <Enhance>-Remove      |                    |
| Delete Row (Multi-row forms)        | Remove                |                    |
| Delete Previous Character           | Delete                |                    |
| Delete Character                    | <Ctrl-D>              |                    |
| Down Arrow                          | Down Arrow            | <Ctrl-J>           |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K> |
| Enhance                             | PF1                   |                    |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Delete   |
| Explain                             | PF2                   |                    |
| GETOUT                              | F10                   | <Enhance>-F        |
| GETOUTALL                           | <Enhance>-F10         |                    |
| Go to Bottom Menu                   | PF3                   |                    |
| Help Menu                           | Help                  |                    |
| Insert Instance (Multi-line fields) | <Enhance>-Insert Here |                    |
| Insert Row (Multi-row forms)        | Insert Here           | <Enhance>-I        |
| Left Arrow                          | Left Arrow            |                    |
| List Choices                        | Select                | <Enhance>-L        |
| Move Window (in Quick-Form)         | <Enhance>-Space       |                    |
| Next Field                          | Tab                   |                    |
| Next Word                           | <Ctrl-F>              |                    |
| Next Screen                         | Next Screen           | <Enhance>-N        |

| Form Action                 | Primary Key           | Alternate Key(s)   |
|-----------------------------|-----------------------|--------------------|
| Previous Screen             | Prev Screen           | <Enhance>-P        |
| Previous Word               | <Ctrl-B>              |                    |
| PREVIOUS                    | PF4                   |                    |
| PROCEED                     | Do                    |                    |
| RETURN                      | Return                |                    |
| Right Arrow                 | Right Arrow           | <Ctrl-K>           |
| SAVE                        | F19                   | <Enhance>-S        |
| Search Current Table        | Find                  | <Enhance>-PF2      |
| Top of Window               | <Enhance>-Up Arrow    | <Enhance>-<Ctrl-U> |
| Top of List                 | <Enhance>-Prev Screen | <Enhance>-T        |
| Typeover/Insert Mode Toggle | <Ctrl-^>              |                    |
| Undo                        | <Ctrl-X>              | <Enhance>-X        |
| Up Arrow                    | Up Arrow              | <Ctrl-U>           |

## Generic

### Generic (pure roll and scroll) Keyboard Mapping

| Form Action                         | Primary Key        | Alternate Key(s) |
|-------------------------------------|--------------------|------------------|
| Beginning of Field                  | <Enhance>-<Ctrl-H> |                  |
| Bottom of List                      |                    |                  |
| Bottom of Window                    | <Enhance>-<Ctrl-J> |                  |
| Break                               |                    |                  |
| Delete Row (Multi-row forms)        | <Ctrl-V>           |                  |
| Delete Instance (Multi-line fields) |                    |                  |
| Delete Word                         |                    |                  |
| Delete Character                    | <Ctrl-D>           |                  |
| Delete Previous Character           | Delete             |                  |
| Down Arrow                          | <Ctrl-J>           |                  |
| End of Field                        | <Enhance>-<Ctrl-K> |                  |
| Enhance                             | <Ctrl-G>           |                  |

---

| Form Action                         | Primary Key        | Alternate Key(s) |
|-------------------------------------|--------------------|------------------|
| Erase Field Line                    | <Ctrl-L>           |                  |
| Explain                             | <Ctrl-F>           |                  |
| GETOUT                              | <Ctrl-Z>           |                  |
| GETOUTALL                           | <Enhance>-<Ctrl-Z> |                  |
| Go to Bottom Menu                   | <Enhance>-<Ctrl-G> |                  |
| Help Menu                           | <Ctrl-A>           |                  |
| Insert Instance (Multi-line fields) |                    |                  |
| Insert Row (Multi-row forms)        |                    |                  |
| Left Arrow                          | <Ctrl-H>           |                  |
| List Choices                        | <Ctrl-O>           |                  |
| Next Word                           |                    |                  |
| Next Screen                         |                    |                  |
| Previous Screen                     |                    |                  |
| Previous Word                       |                    |                  |
| PREVIOUS                            | <Ctrl-P>           |                  |
| PROCEED                             | <Ctrl-N>           |                  |
| RETURN                              | Return             |                  |
| Right Arrow                         | <Ctrl-K>           |                  |
| SAVE                                |                    |                  |
| Search Current Table                | <Ctrl-R>           |                  |
| Tab                                 | <Ctrl-I>           |                  |
| Top of Window                       | <Enhance>-<Ctrl-U> |                  |
| Top of List                         |                    |                  |
| Typeover/Insert Mode Toggle         | <Ctrl-B>           |                  |
| Undo                                |                    |                  |
| Up Arrow                            | <Ctrl-U>           |                  |

**IBM 3151-ANSI**
**IBM 3151 ANSI-like Keyboard Mapping**

| Form Action                         | Primary Key           | Alternate Key(s)    |
|-------------------------------------|-----------------------|---------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  |                     |
| Bottom of List                      | <Enhance>-V           | <Ctrl-V>            |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J>  |
| Break                               | <Ctrl-C>              |                     |
| Delete Word                         | <Ctrl-W>              |                     |
| Delete Instance (Multi-line fields) | <Enhance>-F8          |                     |
| Delete Row (Multi-row forms)        | F8                    | <Enhance>-D         |
| Delete Previous Character           | Backspace             |                     |
| Delete Character                    | <Ctrl-D>              | Del Char            |
| Down Arrow                          | Down Arrow            | <Ctrl-J>            |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K>  |
| Enhance                             | F1                    |                     |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Backspace |
| Explain                             | F2                    |                     |
| Finished Form                       | <Enhance>-F           |                     |
| GETOUT                              | F10                   |                     |
| GETOUTALL                           | <Enhance>-F10         |                     |
| Go to Bottom Window                 | F3                    |                     |
| Help Menu                           | <Enhance>-F3          |                     |
| Insert Instance (Multi-line fields) | <Enhance>-F6          |                     |
| Insert Row (Multi-row forms)        | F11                   | <Enhance>-I         |
| Insert/Typeover/Insert Toggle       | <Enhance>-^           |                     |
| Left Arrow                          | Left Arrow            | Backspace           |
| List Choices                        | F7                    | <Enhance>-L         |
| Macro Ctrl/z                        | <Ctrl-Z>              |                     |
| Macro Definition                    | <Enhance>-K           |                     |

---

| Form Action          | Primary Key        | Alternate Key(s)   |
|----------------------|--------------------|--------------------|
| Macro Gold y         | <Enhance>-Y        |                    |
| Macro Gold z         | <Enhance>-Z        |                    |
| Next Field           | Tab                |                    |
| Next Screen          | <Enhance>-N        |                    |
| Next Word            | <Ctrl-F>           |                    |
| Previous Field       | <Enhance>-Tab      |                    |
| Previous Screen      | <Enhance>-P        |                    |
| Previous Word        | <Ctrl-B>           |                    |
| PREVIOUS             | F4                 |                    |
| PROCEED              | F5                 |                    |
| RETURN               | Return             |                    |
| Right Arrow          | Right Arrow        | <Ctrl-K>           |
| SAVE                 | F9                 | <Enhance>-S        |
| Search Current Table | F6                 |                    |
| Top of Window        | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U> |
| Top of List          | <Enhance>-T        | <Ctrl-T>           |
| Undo                 | <Enhance>-X        | <Ctrl-X>           |
| Up Arrow             | Up Arrow           | <Ctrl-U>           |

## IBM 3151 Ascii Display

### IBM 3151 Ascii Display Station Keyboard Mapping

| Form Action                         | Primary Key          | Alternate Key(s)   |
|-------------------------------------|----------------------|--------------------|
| Beginning of Field                  | <Enhance>-Left Arrow |                    |
| Bottom of List                      | <Enhance>-<- Tab     | <Enhance>-B        |
| Bottom of Window                    | <Enhance>-Down Arrow | <Enhance>-<Ctrl-J> |
| Break                               | <Ctrl-C>             |                    |
| Delete Word                         | <Ctrl-W>             |                    |
| Delete Instance (Multi-line fields) | <Enhance>-F7         |                    |
| Delete Row (Multi-row forms)        | F7                   |                    |

## Appendix B—Keyboard Actions

---

| Form Action                         | Primary Key           | Alternate Key(s)    |
|-------------------------------------|-----------------------|---------------------|
| Delete Previous Character           | Backspace             |                     |
| Delete Character                    | <Ctrl-D>              |                     |
| Down Arrow                          | Down Arrow            | <Ctrl-J>            |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K>  |
| Enhance                             | F1                    |                     |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Backspace |
| Explain                             | F2                    |                     |
| GETOUT                              | F12                   | <Enhance>-F         |
| GETOUTALL                           | <Enhance>-F12         |                     |
| Go to Bottom Menu                   | F3                    |                     |
| Help Menu                           | F5<br><Enhance>-F5    |                     |
| Insert Instance (Multi-line fields) | <Enhance>-F6          |                     |
| Insert Row (Multi-row forms)        | F6                    | <Enhance>-I         |
| Left Arrow                          | Left Arrow            | <Ctrl-H>            |
| List Choices                        | F8                    | <Enhance>-L         |
| Move Window (in Quick-Form)         | <Enhance>-Space       |                     |
| Next Word                           | <Ctrl-F>              |                     |
| Next Screen                         | <- Tab                | <Enhance>-N         |
| Previous Screen                     | Home                  | <Enhance>-P         |
| Previous Word                       | <Ctrl-B>              |                     |
| PREVIOUS                            | F4                    |                     |
| PROCEED                             | F10                   |                     |
| RETURN                              | Return                |                     |
| Right Arrow                         | Right Arrow           | <Ctrl-K>            |
| SAVE                                | F11                   | <Enhance>-S         |
| Search Current Table                | F9                    |                     |
| Tab                                 | Tab                   |                     |
| Top of Window                       | <Enhance>-Up Arrow    | <Enhance>-<Ctrl-U>  |
| Top of List                         | <Enhance>-Home        | <Enhance>-T         |



---

| Form Action                 | Primary Key | Alternate Key(s) |
|-----------------------------|-------------|------------------|
| Typeover/Insert Mode Toggle | <Enhance>-^ |                  |
| Undo                        | <Ctrl-X>    | <Enhance>-X      |
| Up Arrow                    | Up Arrow    | <Ctrl-U>         |

## IBM 6091

### IBM 6091 Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)                         |
|-------------------------------------|-----------------------|------------------------------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  | <Enhance>-<Ctrl-H>                       |
| Bottom of List                      | <Enhance>-Page Down   | <Enhance>-End                            |
| Bottom of Window                    | End                   | <Enhance>-<Ctrl-J><br><Enhance>-<Ctrl-V> |
| Break                               | <Ctrl-C>              |                                          |
| Delete Row (Multi-row forms)        | F8                    |                                          |
| Delete Instance (Multi-line fields) |                       |                                          |
| Delete Word                         | <Ctrl-W>              | <Enhance>-W                              |
| Delete Character                    | Del                   | <Ctrl-D>                                 |
| Delete Previous Character           | Delete                | <Ctrl-H><br><-Backspace                  |
| Down Arrow                          | Down Arrow            | <Ctrl-J><br><Ctrl-V>                     |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K>                       |
| Enhance                             | F1                    | <Ctrl-G>                                 |
| Erase Field Line                    | <Ctrl-L>              |                                          |
| Explain                             | F2                    |                                          |
| GETOUT                              | F10                   |                                          |
| GETOUTALL                           | <Enhance>-F10         |                                          |
| Go to Bottom Menu                   | F3                    |                                          |
| Go to Top Menu                      | <Enhance>-F1          |                                          |
| Help Menu                           | <Enhance>-F3          |                                          |

## Appendix B—Keyboard Actions

---

| Form Action                         | Primary Key        | Alternate Key(s)   |
|-------------------------------------|--------------------|--------------------|
| Insert Instance (Multi-line fields) | <Enhance>-Insert   |                    |
| Insert Row (Multi-row forms)        | Insert             |                    |
| Left Arrow                          | Left Arrow         | <Ctrl-H>           |
| List Choices                        | F7                 | <Enhance>-F2       |
| Macro Definition                    | <Enhance>-K        |                    |
| Next Field                          | Tab                |                    |
| Next Screen                         | Page Down          |                    |
| Next Word                           | <Ctrl-F>           | <Enhance>-F        |
| Previous Field                      | <Enhance>-Tab      |                    |
| Previous Screen                     | Page Up            |                    |
| Previous Word                       | <Ctrl-B>           | <Enhance>-B        |
| PREVIOUS                            | F4                 |                    |
| PROCEED                             | F5                 |                    |
| RETURN                              | Return             |                    |
| Right Arrow                         | Right Arrow        | <Ctrl-K>           |
| SAVE                                | F9                 |                    |
| Search Current Table                | F6                 |                    |
| Top of Window                       | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U> |
| Top of List                         | <Enhance>-Page Up  | <Enhance>-Home     |
| Typeover/Insert Mode Toggle         | <Ctrl-^>           |                    |
| Undo                                | <Enhance>-X        | <Ctrl-X>           |
| Up Arrow                            | Up Arrow           | <Ctrl-U>           |

## IBM PC

### IBM PC Keyboard Mapping

| Form Action        | Primary Key          | Alternate Key(s) |
|--------------------|----------------------|------------------|
| Beginning of Field | <Enhance>-Left Arrow |                  |
| Bottom of List     | <Enhance>-PageDn     |                  |

| Form Action                         | Primary Key           | Alternate Key(s)                      |
|-------------------------------------|-----------------------|---------------------------------------|
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J><br><Ctrl-V>        |
| Break                               | <Ctrl-C>              |                                       |
| Delete Word                         | <Ctrl-W>              | <Enhance>-W                           |
| Delete Instance (Multi-line fields) | <Enhance>-F8          |                                       |
| Delete Row (Multi-row forms)        | F8                    |                                       |
| Delete Previous Character           | <- (Previous Space)   | <Ctrl-H>                              |
| Delete Character                    | Delete                | <Ctrl-D><br><Enhance>-D               |
| Down Arrow                          | Down Arrow            | <Ctrl-J>                              |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K>                    |
| Enhance                             | F1                    | <Ctrl-G>                              |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Del<br><Enhance>-Delete Key |
| Explain                             | F2                    |                                       |
| GETOUT                              | F10                   |                                       |
| GETOUTALL                           | <Enhance>-F10         |                                       |
| Go to Bottom Menu                   | F3                    |                                       |
| Help Menu                           | <Enhance>-F3          |                                       |
| Insert Instance (Multi-line fields) | <Enhance>-Ins         |                                       |
| Insert Row (Multi-row forms)        | Ins                   | <Enhance>-I                           |
| Left Arrow                          | Left Arrow            |                                       |
| List Choices                        | F7                    | <Enhance>-F2                          |
| Next Field                          | Tab                   |                                       |
| Next Word                           | <Ctrl-F>              | <Enhance>-F                           |
| Next Screen                         | PgDn                  |                                       |
| Previous Screen                     | PgUp                  |                                       |
| Previous Word                       | <Ctrl-B>              | <Enhance>-B                           |
| PREVIOUS                            | F4                    |                                       |
| PROCEED                             | F5                    |                                       |
| RETURN                              | Return                |                                       |

| Form Action                 | Primary Key        | Alternate Key(s)   |
|-----------------------------|--------------------|--------------------|
| Right Arrow                 | Right Arrow        | <Ctrl-K>           |
| SAVE                        | F9                 |                    |
| Search Current Table        | F6                 |                    |
| Top of Window               | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U> |
| Top of List                 | <Enhance>-PgUp     |                    |
| Typeover/Insert Mode Toggle | <Ctrl-^>           |                    |
| Undo                        | <Ctrl-X>           | <Enhance>-X        |
| Up Arrow                    | Up Arrow           | <Ctrl-U>           |

## IBM PC With Color

### IBM PC with wired Color Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)                                |
|-------------------------------------|-----------------------|-------------------------------------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  |                                                 |
| Bottom of List                      | <Enhance>-PageDn      | <Enhance>-End                                   |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J><br><Enhance>-<Ctrl-V><br>End |
| Break                               | <Ctrl-C>              |                                                 |
| Delete Word                         | <Ctrl-W>              | <Enhance>-W                                     |
| Delete Instance (Multi-line fields) | <Enhance>-F8          |                                                 |
| Delete Row (Multi-row forms)        | F8                    |                                                 |
| Delete Previous Character           | <- (Previous Space)   | Delete<br><Ctrl-H>                              |
| Delete Character                    | Del                   | <Ctrl-D><br><Enhance>-D                         |
| Down Arrow                          | Down Arrow            | <Ctrl-J><br><Ctrl-V>                            |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K>                              |
| Enhance                             | F1                    | <Ctrl-G>                                        |

---

| Form Action                         | Primary Key        | Alternate Key(s)                      |
|-------------------------------------|--------------------|---------------------------------------|
| Erase Field Line                    | <Ctrl-L>           | <Enhance>-Del<br><Enhance>-Delete Key |
| Explain                             | F2<br><Enhance>-F2 |                                       |
| GETOUT                              | F10                |                                       |
| GETOUTALL                           | <Enhance>-F10      |                                       |
| Go to Bottom Menu                   | F3                 |                                       |
| Help Menu                           | <Enhance>-F3       |                                       |
| Insert Instance (Multi-line fields) | <Enhance>-Ins      |                                       |
| Insert Row (Multi-row forms)        | Ins                | <Enhance>-I                           |
| Left Arrow                          | Left Arrow         |                                       |
| List Choices                        | F7                 | <Enhance>-F2                          |
| Next Field                          | Tab                |                                       |
| Next Word                           | <Ctrl-F>           | <Enhance>-F                           |
| Next Screen                         | PgDn               |                                       |
| Previous Screen                     | PgUp               |                                       |
| Previous Word                       | <Ctrl-B>           | <Enhance>-B                           |
| PREVIOUS                            | F4                 |                                       |
| PROCEED                             | F5                 |                                       |
| RETURN                              | Return             |                                       |
| Right Arrow                         | Right Arrow        | <Ctrl-K>                              |
| SAVE                                | F9                 |                                       |
| Search Current Table                | F6                 |                                       |
| Top of Window                       | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U>                    |
| Top of List                         | <Enhance>-PgUp     | <Enhance>-Home                        |
| Typeover Mode Toggle                | <Enhance>-^        |                                       |
| Undo                                | <Ctrl-X>           | <Enhance>-X                           |
| Up Arrow                            | Up Arrow           | <Ctrl-U>                              |

## MSM PC Console

### MSM PC Console Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)    |
|-------------------------------------|-----------------------|---------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  |                     |
| Bottom of List                      | <Enhance>-Page Down   | <Enhance>-B         |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J>  |
| Break                               | <Ctrl-C>              |                     |
| Delete Row (Multi-row forms)        | Delete                |                     |
| Delete Instance (Multi-line fields) | <Enhance>-Delete      |                     |
| Delete Word                         | <Ctrl-W>              |                     |
| Delete Character                    | <Ctrl-D>              |                     |
| Delete Previous Character           | Backspace             |                     |
| Down Arrow                          | Down Arrow            | <Ctrl-J>            |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K>  |
| Enhance                             | F1                    |                     |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Backspace |
| Explain                             | F2                    |                     |
| GETOUT                              | F10                   | <Enhance>-F         |
| GETOUTALL                           | <Enhance>-F10         |                     |
| Go to Bottom Menu                   | F3                    |                     |
| Help Menu                           | <Enhance>-F3          |                     |
| Insert Instance (Multi-line fields) | <Enhance>-Insert      |                     |
| Insert Row (Multi-row forms)        | Insert                | <Enhance>-I         |
| Left Arrow                          | Left Arrow            |                     |
| List Choices                        | F7                    | <Enhance>-L         |
| Next Field                          | Tab                   |                     |
| Next Word                           | <Ctrl-F>              |                     |
| Next Screen                         | Page Down             | <Enhance>-N         |
| Previous Screen                     | Page Up               | <Enhance>-P         |
| Previous Word                       | <Ctrl-B>              |                     |

---

| Form Action                 | Primary Key        | Alternate Key(s) |
|-----------------------------|--------------------|------------------|
| PREVIOUS                    | F4                 |                  |
| PROCEED                     | F8                 |                  |
| RETURN                      | Return             |                  |
| Right Arrow                 | Right Arrow        | <Ctrl-K>         |
| SAVE                        | F9                 | <Enhance>-S      |
| Search Current Table        | F6                 | <Enhance>-F2     |
| Top of Window               | <Enhance>-Up Arrow |                  |
| Top of List                 | <Enhance>-Page Up  | <Enhance>-T      |
| Typeover/Insert Mode Toggle | <Enhance>-^        |                  |
| Undo                        | <Ctrl-X>           | <Enhance>-X      |
| Up Arrow                    | Up Arrow           | <Ctrl-U>         |

## NT Console

### NT Console (DTM-PC Console) Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)   |
|-------------------------------------|-----------------------|--------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  |                    |
| Bottom of List                      | <Enhance>-Page Down   | <Enhance>-B        |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J> |
| Break                               | <Ctrl-C>              |                    |
| Delete Row (Multi-row forms)        | Delete                | <Enhance>-D        |
| Delete Instance (Multi-line fields) | <Enhance>-Delete      |                    |
| Delete Character                    | <Ctrl-D>              |                    |
| Delete Previous Character           | Backspace             |                    |
| Delete Word                         | <Ctrl-W>              |                    |
| Down Arrow                          | Down Arrow            | <Ctrl-J>           |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K> |
| Enhance                             | F1                    | <Ctrl-G>           |
| Erase Field Line                    | <Ctrl-L>              |                    |
| Explain                             | F2                    |                    |

## Appendix B—Keyboard Actions

---

| Form Action                         | Primary Key        | Alternate Key(s) |
|-------------------------------------|--------------------|------------------|
| GETOUT                              | F10                | <Enhance>-F      |
| GETOUTALL                           | <Enhance>-F10      |                  |
| Go to Bottom Menu                   | F3                 |                  |
| Help Menu                           | F3                 |                  |
| Insert Instance (Multi-line fields) | <Enhance>-Insert   |                  |
| Insert Row (Multi-row forms)        | Insert             | <Enhance>-I      |
| Left Arrow                          | Left Arrow         |                  |
| List Choices                        | F7                 |                  |
| Next Field                          | Tab                |                  |
| Next Word                           | <Ctrl-F>           |                  |
| Next Screen                         | Page Down          | <Enhance>-N      |
| Previous Screen                     | Page Up            | <Enhance>-P      |
| Previous Word                       | <Ctrl-B>           |                  |
| PREVIOUS                            | F4                 |                  |
| PROCEED                             | F8                 |                  |
| RETURN                              | Enter              |                  |
| Right Arrow                         | Right Arrow        | <Ctrl-K>         |
| SAVE                                | F9                 | <Enhance>-S      |
| Search Current Table                | F6                 |                  |
| Top of Window                       | <Enhance>-Up Arrow |                  |
| Top of List                         | <Enhance>-Page Up  | <Enhance>-T      |
| Typeover/Insert Mode Toggle         | <Ctrl-^>           |                  |
| Undo                                | <Enhance>-X        |                  |
| Up Arrow                            | Up Arrow           |                  |



---

## Open M Terminal

### Open M Terminal Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)    |
|-------------------------------------|-----------------------|---------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  |                     |
| Bottom of List                      | <Enhance>-B           | <Enhance>-Page Down |
| Bottom of Window                    | <Enhance>-Down Arrow  |                     |
| Break                               | <Ctrl-C>              |                     |
| Delete Row (Multi-row forms)        | Delete                | <Enhance>-D         |
| Delete Instance (Multi-line fields) | <Enhance>-Delete      |                     |
| Delete Word                         | <Ctrl-W>              |                     |
| Delete Character                    | <Ctrl-D>              |                     |
| Delete Previous Character           | Backspace             |                     |
| Down Arrow                          | Down Arrow            | <Ctrl-J>            |
| End of Field                        | <Enhance>-Right Arrow |                     |
| Enhance                             | F1                    | <Ctrl-G>            |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Backspace |
| Explain                             | F2                    |                     |
| GETOUT                              | F10                   | <Enhance>-F         |
| GETOUTALL                           | <Enhance>-F10         |                     |
| Go to Bottom Menu                   | <Enhance>-F1          |                     |
| Help Menu                           | <Enhance>-F3          |                     |
| Insert Instance (Multi-line fields) | <Enhance>-Insert      |                     |
| Insert Row (Multi-row forms)        | Insert                | <Enhance>-I         |
| Left Arrow                          | Left Arrow            | <Ctrl-H>            |
| List Choices                        | F7                    |                     |
| Macro Definition                    | <Enhance>-K           |                     |
| Next Field                          | Tab                   |                     |
| Next Word                           | <Ctrl-F>              |                     |
| Next Screen                         | Page Down             | <Enhance>-N         |
| Previous Screen                     | Page Up               | <Enhance>-P         |

## Appendix B—Keyboard Actions

---

| Form Action                 | Primary Key        | Alternate Key(s)  |
|-----------------------------|--------------------|-------------------|
| Previous Word               | <Ctrl-B>           |                   |
| PREVIOUS                    | F4                 | <Enhance>-Tab     |
| PROCEED                     | F8                 |                   |
| RETURN                      | Return             |                   |
| Right Arrow                 | Right Arrow        | <Ctrl-K>          |
| SAVE                        | F9                 | <Enhance>-S       |
| Search Current Table        | F6                 |                   |
| Top of Window               | <Enhance>-Up Arrow |                   |
| Top of List                 | <Enhance>-T        | <Enhance>-Page Up |
| Typeover/Insert Mode Toggle | <Ctrl-^>           |                   |
| Undo                        | <Enhance>-X        |                   |
| Up Arrow                    | Up Arrow           | <Ctrl-U>          |

## PC Console

### PC Console (DESQView) Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)        |
|-------------------------------------|-----------------------|-------------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  |                         |
| Bottom of List                      | <Enhance>-Page Down   | <Enhance>-V<br><Ctrl-V> |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J>      |
| Break                               | <Ctrl-C>              |                         |
| Delete Word                         | <Ctrl-W>              | <Enhance>-W             |
| Delete Instance (Multi-line fields) | <Enhance>-F8          |                         |
| Delete Row (Multi-row forms)        | F8                    |                         |
| Delete Previous Character           | <--                   |                         |
| Delete Character                    | Delete                | <Ctrl-D><br><Enhance>-D |
| Down Arrow                          | Down Arrow            | <Ctrl-J>                |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K>      |

| Form Action                         | Primary Key        | Alternate Key(s)                  |
|-------------------------------------|--------------------|-----------------------------------|
| Enhance                             | F1                 | <Ctrl-G>                          |
| Erase Field Line                    | <Ctrl-L>           | <Enhance>-Delete<br><Enhance>-<-- |
| Explain                             | F2                 |                                   |
| GETOUT                              | F10                |                                   |
| GETOUTALL                           | <Enhance>-F10      |                                   |
| Go to Bottom Menu                   | F3                 |                                   |
| Help Menu                           | F3                 |                                   |
| Insert Instance (Multi-line fields) | <Enhance>-Insert   |                                   |
| Insert Row (Multi-row forms)        | Insert             | <Enhance>-I                       |
| Left Arrow                          | Left Arrow         |                                   |
| List Choices                        | F7                 |                                   |
| Next Field                          | Tab                |                                   |
| Next Word                           | <Ctrl-F>           | <Enhance>-F                       |
| Next Screen                         | Page Down          | <Enhance>-N                       |
| Previous Screen                     | Page Up            | <Enhance>-P                       |
| Previous Word                       | <Ctrl-B>           | <Enhance>-B                       |
| PREVIOUS                            | F4                 |                                   |
| PROCEED                             | F5                 |                                   |
| RETURN                              | Return             |                                   |
| Right Arrow                         | Right Arrow        | <Ctrl-K>                          |
| SAVE                                | F9                 |                                   |
| Search Current Table                | F6                 | <Enhance>-F2                      |
| Top of Window                       | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U>                |
| Top of List                         | <Enhance>-Page Up  | <Enhance>-T<br><Ctrl-T>           |
| Typeover/Insert Mode Toggle         | <Ctrl-^>           |                                   |
| Undo                                | <Ctrl-X>           | <Enhance>-X                       |
| Up Arrow                            | Up Arrow           | <Ctrl-U>                          |

# SUN

## SUN Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)                           |
|-------------------------------------|-----------------------|--------------------------------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  | <Enhance>-<Ctrl-H>                         |
| Bottom of List                      | <Enhance>-Page Down   | <Enhance>-End<br>Ctrl-V                    |
| Bottom of Window                    | End                   | <Enhance>-Down Arrow<br><Enhance>-<Ctrl-J> |
| Break                               | <Ctrl-C>              |                                            |
| Delete Row (Multi-row forms)        | F8                    |                                            |
| Delete Instance (Multi-line fields) | <Enhance>-Del Line    |                                            |
| Delete Word                         | <Ctrl-W>              |                                            |
| Delete Character                    | Del                   | <Ctrl-D>                                   |
| Delete Previous Character           | Backspace             | Delete                                     |
| Down Arrow                          | Down Arrow            | <Ctrl-J>                                   |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K>                         |
| Enhance                             | F1                    | <Ctrl-G>                                   |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Del<br><Enhance>-Delete          |
| Explain                             | F2                    |                                            |
| GETOUT                              | F10                   | <Enhance>-F                                |
| GETOUTALL                           | <Enhance>-F10         |                                            |
| Go to Bottom Menu                   | F3                    |                                            |
| Help Menu                           | Help                  | <Enhance>-F3                               |
| Insert Instance (Multi-line fields) | <Enhance>-Insert      |                                            |
| Insert Row (Multi-row forms)        | Insert                | <Enhance>-I                                |
| Left Arrow                          | Left Arrow            | <Ctrl-H>                                   |
| List Choices                        | F7                    |                                            |
| Macro Definition                    | <Enhance>-K           |                                            |
| Next Field                          | Tab                   |                                            |
| Next Word                           | <Ctrl-F>              |                                            |

---

| Form Action                 | Primary Key        | Alternate Key(s)           |
|-----------------------------|--------------------|----------------------------|
| Next Screen                 | Page Down          | <Enhance>-N                |
| Previous Screen             | Page Up            | <Enhance>-P                |
| Previous Word               | <Ctrl-B>           | <Enhance>-B                |
| PREVIOUS                    | F4                 |                            |
| PROCEED                     | F5                 |                            |
| RETURN                      | Return             |                            |
| Right Arrow                 | Right Arrow        | <Ctrl-K>                   |
| SAVE                        | F9                 |                            |
| Search Current Table        | F6                 |                            |
| Top of Window               | <Enhance>-Up Arrow | Home<br><Enhance>-<Ctrl-U> |
| Top of List                 | <Enhance>-Page Up  | <Enhance>-Home<br><Ctrl-T> |
| Typeover/Insert Mode Toggle | <Ctrl-^>           |                            |
| Undo                        | <Ctrl-X>           |                            |
| Up Arrow                    | Up Arrow           | <Ctrl-U>                   |

## TV905

### Televideo 905 Terminal Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s) |
|-------------------------------------|-----------------------|------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  |                  |
| Bottom of List                      | <Enhance>-V           |                  |
| Bottom of Window                    | <Enhance>-Down Arrow  |                  |
| Break                               | <Ctrl-C>              |                  |
| Delete Row (Multi-row forms)        | Line Delete           |                  |
| Delete Instance (Multi-line fields) | <Enhance>-Line Delete |                  |
| Delete Word                         | <Ctrl-W>              | <Enhance>-W      |
| Delete Character                    | <Ctrl-D>              | <Enhance>-D      |
| Delete Previous Character           | Del                   |                  |

## Appendix B—Keyboard Actions

---

| Form Action                         | Primary Key           | Alternate Key(s)       |
|-------------------------------------|-----------------------|------------------------|
| Down Arrow                          | Down Arrow            | <Ctrl-V>               |
| End of Field                        | <Enhance>-Right Arrow |                        |
| Enhance                             | F1                    | <Ctrl-G>               |
| Erase Field Line                    | <Enhance>-Del         | <Ctrl-U><br>Line Erase |
| Explain                             | F2                    |                        |
| GETOUT                              | <Enhance>-F           |                        |
| GETOUTALL                           | <Enhance>-F4          |                        |
| Go to Bottom Menu                   | F3                    |                        |
| Help Menu                           | <Enhance>-F3          |                        |
| Insert Instance (Multi-line fields) | <Enhance>-Line Insert |                        |
| Insert Row (Multi-row forms)        | Line Insert           | <Enhance>-I            |
| Left Arrow                          | Left Arrow            |                        |
| List Choices                        | <Enhance>-L           |                        |
| Next Field                          | Tab                   |                        |
| Next Word                           | <Ctrl-F>              | <Enhance>-F            |
| Next Screen                         | <Enhance>-N           |                        |
| Previous Field                      | <Ctrl-P>              |                        |
| Previous Screen                     | <Enhance>-P           |                        |
| Previous Word                       | <Ctrl-B>              | <Enhance>-B            |
| PREVIOUS                            | F4                    |                        |
| PROCEED                             | Esc-Esc               |                        |
| RETURN                              | Return                |                        |
| Right Arrow                         | Right Arrow           | <Ctrl-L>               |
| Search Current Table                | <Enhance>-F2          |                        |
| Top of Window                       | <Enhance>-Up Arrow    |                        |
| Top of List                         | <Enhance>-T           | <Ctrl-T>               |
| Typeover/Insert Mode Toggle         | <Ctrl-^>              |                        |
| Undo                                | <Enhance>-X           | <Ctrl-X>               |
| Up Arrow                            | Up Arrow              | <Ctrl-K>               |

---

## Unisys Console

### Unisys Console Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)                     |
|-------------------------------------|-----------------------|--------------------------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  | ^E Left Arrow                        |
| Bottom of List                      | <Enhance>-Page Down   | <Enhance>-V<br><Ctrl-V>              |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J><br>^E Down Arrow  |
| Break                               | <Ctrl-C>              |                                      |
| Delete Row (Multi-row forms)        | F8                    |                                      |
| Delete Instance (Multi-line fields) | <Enhance>-F8          |                                      |
| Delete Word                         | <Ctrl-W>              | <Enhance>-W                          |
| Delete Character                    | Delete                | <Ctrl-D><br><Enhance>-D              |
| Delete Previous Character           | <--                   |                                      |
| Down Arrow                          | Down Arrow            | <Ctrl-J>                             |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K><br>^E Right Arrow |
| Enhance                             | F1                    | <Ctrl-G>                             |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Delete<br><Enhance>-<--    |
| Explain                             | F2                    |                                      |
| GETOUT                              | F10                   |                                      |
| GETOUTALL                           | <Enhance>-F10         |                                      |
| Go to Bottom Menu                   | F3                    |                                      |
| Go to Top Menu                      | <Enhance>-F1          |                                      |
| Help Menu                           | <Enhance>-F3          |                                      |
| Insert Instance (Multi-line fields) | <Enhance>-Insert      |                                      |
| Insert Row (Multi-row forms)        | Insert                | <Enhance>-I                          |
| Left Arrow                          | Left Arrow            |                                      |
| List Choices                        | F7                    |                                      |

## Appendix B—Keyboard Actions

---

| Form Action                 | Primary Key        | Alternate Key(s)                  |
|-----------------------------|--------------------|-----------------------------------|
| Macro Definition            | <Enhance>-K        |                                   |
| Next Field                  | Tab                |                                   |
| Next Screen                 | Page Down          | <Enhance>-N                       |
| Next Word                   | <Ctrl-F>           | <Enhance>-F                       |
| Previous Field              | <Enhance>-Tab      |                                   |
| Previous Screen             | Page Up            | <Enhance>-P                       |
| Previous Word               | <Ctrl-B>           | <Enhance>-B                       |
| PREVIOUS                    | F4                 |                                   |
| PROCEED                     | F5                 |                                   |
| RETURN                      | Return             |                                   |
| Right Arrow                 | Right Arrow        | <Ctrl-K>                          |
| SAVE                        | F9                 |                                   |
| Search Current Table        | F6                 | <Enhance>-F2                      |
| Top of Window               | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U><br>^E Up Arrow |
| Top of List                 | <Ctrl-T>           | <Enhance>-T<br><Enhance>-Page Up  |
| Typeover/Insert Mode Toggle | <Ctrl-^>           |                                   |
| Undo                        | <Ctrl-X>           | <Enhance>-X                       |
| Up Arrow                    | Up Arrow           | <Ctrl-U>                          |

## VT100

### DEC VT 100 Keyboard Mapping

| Form Action                  | Primary Key          | Alternate Key(s)   |
|------------------------------|----------------------|--------------------|
| Beginning of Field           | <Enhance>-Left Arrow | <Enhance>-<Ctrl-H> |
| Bottom of List               | <Enhance>-B          |                    |
| Bottom of Window             | <Enhance>-Down Arrow | <Enhance>-<Ctrl-J> |
| Break                        | <Ctrl-C>             |                    |
| Delete Row (Multi-row forms) | <Enhance>-D          |                    |



| Form Action                         | Primary Key           | Alternate Key(s)    |
|-------------------------------------|-----------------------|---------------------|
| Delete Instance (Multi-line fields) | <Enhance>-Delete      |                     |
| Delete Word                         | <Ctrl-W>              |                     |
| Delete Character                    | <Ctrl-D>              |                     |
| Delete Previous Character           | Delete                |                     |
| Down Arrow                          | Down Arrow            | <Ctrl-J>            |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K>  |
| Enhance                             | PF1                   | <Ctrl-G>            |
| Erase Field Line                    | <Ctrl-L>              |                     |
| Explain                             | PF2                   |                     |
| GETOUT                              | <Enhance>-F           |                     |
| GETOUTALL                           | <Enhance>-PF4         |                     |
| Go to Bottom Menu                   | PF3                   |                     |
| Go to Top Menu                      | <Enhance>-PF1         |                     |
| Help Menu                           | <Enhance>-PF3         |                     |
| Insert Instance (Multi-line fields) | <Enhance>-G           |                     |
| Insert Row (Multi-row forms)        | <Enhance>-I           |                     |
| Left Arrow                          | Left Arrow            | <Ctrl-H>            |
| List Choices                        | <Enhance>-L           |                     |
| Macro Definition                    | <Enhance>-K           |                     |
| Next Field                          | Tab                   |                     |
| Next Screen                         | <Enhance>-N           |                     |
| Next Word                           | <Ctrl-F>              |                     |
| Previous Field                      | <Enhance>-Tab         |                     |
| Previous Screen                     | <Enhance>-P           |                     |
| Previous Word                       | <Ctrl-B>              |                     |
| PREVIOUS                            | PF4                   |                     |
| PROCEED                             | <Enhance>-C           | <Enhance>-Space Bar |
| RETURN                              | Return                |                     |
| Right Arrow                         | Right Arrow           | <Ctrl-K>            |
| SAVE                                | <Enhance>-S           |                     |

| Form Action                 | Primary Key        | Alternate Key(s)   |
|-----------------------------|--------------------|--------------------|
| Search Current Table        | <Enhance>-PF2      |                    |
| Top of Window               | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U> |
| Top of List                 | <Enhance>-T        |                    |
| Typeover/Insert Mode Toggle | <Ctrl-^>           |                    |
| Undo                        | <Enhance>-X        |                    |
| Up Arrow                    | Up Arrow           | <Ctrl-U>           |

## VT220

### DEC VT 220 and Compatables Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)   |
|-------------------------------------|-----------------------|--------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  | <Enhance>-<Ctrl-H> |
| Bottom of List                      | <Enhance>-Next Screen | <Enhance>-B        |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J> |
| Break                               | <Ctrl-C>              |                    |
| Delete Word                         | <Ctrl-W>              |                    |
| Delete Instance (Multi-line fields) | <Enhance>-Remove      |                    |
| Delete Row (Multi-row forms)        | Remove                |                    |
| Delete Previous Character           | Delete                |                    |
| Delete Character                    | <Ctrl-D>              |                    |
| Down Arrow                          | Down Arrow            | <Ctrl-J>           |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K> |
| Enhance                             | PF1                   |                    |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Delete   |
| Explain                             | PF2                   |                    |
| GETOUT                              | F20                   |                    |
| GETOUTALL                           | <Enhance>-F20         |                    |
| Go to Bottom Menu                   | PF3                   |                    |
| Help Menu                           | Help                  |                    |

| Form Action                         | Primary Key           | Alternate Key(s)   |
|-------------------------------------|-----------------------|--------------------|
| Insert Instance (Multi-line fields) | <Enhance>-Insert Here |                    |
| Insert Row (Multi-row forms)        | Insert Here           | <Enhance>-I        |
| Left Arrow                          | Left Arrow            | <Ctrl-H>           |
| List Choices                        | Select                | <Enhance>-L        |
| Move Window (in Quick-Form)         | <Enhance>-Space       |                    |
| Next Field                          | Tab                   |                    |
| Next Word                           | <Ctrl-F>              |                    |
| Next Screen                         | Next Screen           | <Enhance>-N        |
| Previous Screen                     | Prev Screen           | <Enhance>-P        |
| Previous Word                       | <Ctrl-B>              |                    |
| PREVIOUS                            | PF4                   |                    |
| PROCEED                             | Do                    |                    |
| RETURN                              | Return                |                    |
| Right Arrow                         | Right Arrow           | <Ctrl-K>           |
| SAVE                                | F19                   | <Enhance>-S        |
| Search Current Table                | Find                  | <Enhance>-PF2      |
| Top of Window                       | <Enhance>-Up Arrow    | <Enhance>-<Ctrl-U> |
| Top of List                         | <Enhance>-Prev Screen | <Enhance>-T        |
| Typeover/Insert Mode Toggle         | <Enhance>-^           |                    |
| Undo                                | <Ctrl-X>              | <Enhance>-X        |
| Up Arrow                            | Up Arrow              | <Ctrl-U>           |

## WAYTEC

### WAYTEC (tis-pc) Keyboard Mapping

| Form Action        | Primary Key          | Alternate Key(s)                   |
|--------------------|----------------------|------------------------------------|
| Beginning of Field | <Enhance>-Left Arrow | ^E Left Arrow                      |
| Bottom of List     | <Ctrl-V>             | <Enhance>-V<br><Enhance>-Page Down |

## Appendix B—Keyboard Actions

---

| Form Action                         | Primary Key           | Alternate Key(s)                     |
|-------------------------------------|-----------------------|--------------------------------------|
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J><br>^E Down Arrow  |
| Break                               | <Ctrl-C>              |                                      |
| Delete Row (Multi-row forms)        | F8                    |                                      |
| Delete Instance (Multi-line fields) | <Enhance>-F8          |                                      |
| Delete Word                         | <Ctrl-W>              | <Enhance>-W                          |
| Delete Character                    | Delete                | <Ctrl-D><br><Enhance>-D              |
| Delete Previous Character           | <--                   |                                      |
| Down Arrow                          | Down Arrow            | <Ctrl-J>                             |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K><br>^E Right Arrow |
| Enhance                             | F1                    | <Ctrl-G>                             |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Delete<br><Enhance>-<--    |
| Explain                             | F2                    |                                      |
| GETOUT                              | F10                   |                                      |
| GETOUTALL                           | <Enhance>-F10         |                                      |
| Go to Bottom Menu                   | F3                    |                                      |
| Go to Top Menu                      | <Enhance>-F1          |                                      |
| Help Menu                           | <Enhance>-F3          |                                      |
| Insert Instance (Multi-line fields) | <Enhance>-Insert      |                                      |
| Insert Row (Multi-row forms)        | Insert                | <Enhance>-I                          |
| Left Arrow                          | Left Arrow            |                                      |
| List Choices                        | F7                    |                                      |
| Macro Definition                    | <Enhance>-K           |                                      |
| Next Field                          | Tab                   |                                      |
| Next Screen                         | Page Down             | <Enhance>-N                          |
| Next Word                           | <Ctrl-F>              | <Enhance>-F                          |
| Previous Field                      | <Enhance>-Tab         |                                      |
| Previous Screen                     | Page Up               | <Enhance>-P                          |

---

| Form Action                 | Primary Key        | Alternate Key(s)                  |
|-----------------------------|--------------------|-----------------------------------|
| Previous Word               | <Ctrl-B>           | <Enhance>-B                       |
| PREVIOUS                    | F4                 |                                   |
| PROCEED                     | F5                 |                                   |
| RETURN                      | Return             |                                   |
| Right Arrow                 | Right Arrow        | <Ctrl-K>                          |
| SAVE                        | F9                 |                                   |
| Search Current Table        | F6                 | <Enhance>-F2                      |
| Top of Window               | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U><br>^E Up Arrow |
| Top of List                 | <Ctrl-T>           | <Enhance>-T<br><Enhance>-Page Up  |
| Typeover/Insert Mode Toggle | <Ctrl-^>           |                                   |
| Undo                        | <Ctrl-X>           | <Enhance>-X                       |
| Up Arrow                    | Up Arrow           | <Ctrl-U>                          |

## WYSE60 (Native)

### WYSE60 (Native Mode) Keyboard Mapping

| Form Action                         | Primary Key          | Alternate Key(s)                |
|-------------------------------------|----------------------|---------------------------------|
| Beginning of Field                  | <Enhance>-Left Arrow | <Enhance>-<Ctrl-H><br>Backspace |
| Bottom of List                      | <Enhance>-Page Down  | <Enhance>-V<br>End              |
| Bottom of Window                    | <Enhance>-Down Arrow | <Enhance>-<Ctrl-J>              |
| Break                               | <Ctrl-C>             |                                 |
| Delete Word                         | <Ctrl-W>             |                                 |
| Delete Character                    | <Ctrl-D>             | Del Char                        |
| Delete Previous Character           | Del                  |                                 |
| Delete Instance (Multi-line fields) | <Enhance>-PF8        |                                 |
| Delete Row (Multi-row forms)        | PF8                  | Delete<br><Enhance>-D           |

## Appendix B—Keyboard Actions

---

| Form Action                         | Primary Key           | Alternate Key(s)      |
|-------------------------------------|-----------------------|-----------------------|
| Delete Previous Character           | Delete                |                       |
| Down Arrow                          | Down Arrow            | <Ctrl-J>              |
| End of Field                        | <Enhance>-Right Arrow |                       |
| Enhance                             | PF1                   | <Ctrl-G>              |
| Erase Field Line                    | PF12                  | <Enhance>-Del         |
| Explain                             | PF2                   |                       |
| GETOUT                              | PF10                  | <Enhance>-F           |
| GETOUTALL                           | <Enhance>-F10         | <Enhance>-PF4         |
| Go to Bottom Menu                   | PF3                   |                       |
| Help Menu                           | <Enhance>-PF3         |                       |
| Insert Instance (Multi-line fields) | <Enhance>-Insert      | <Enhance>-G           |
| Insert Row (Multi-row forms)        | Insert                | <Enhance>-I           |
| Insert Mode Toggle                  | Ins                   | <Enhance>-Ins         |
| Left Arrow                          | Left Arrow            | <Ctrl-H><br>Backspace |
| List Choices                        | PF7                   | <Enhance>-L           |
| Macro Ctrl/z                        | <Ctrl-Z>              |                       |
| Macro Definition                    | <Enhance>-K           |                       |
| Macro Gold y                        | <Enhance>-Y           |                       |
| Macro Gold z                        | <Enhance>-Z           |                       |
| Next Field                          | Tab                   |                       |
| Next Word                           | <Ctrl-F>              |                       |
| Next Screen                         | Page Down             | <Enhance>-N           |
| Previous Screen                     | Page Up               | <Enhance>-P           |
| Previous Word                       | <Ctrl-B>              |                       |
| PREVIOUS                            | PF4                   |                       |
| PROCEED                             | PF5                   | Esc-Esc               |
| RETURN                              | Return                |                       |
| Right Arrow                         | Right Arrow           | <Ctrl-L>              |
| SAVE                                | PF9                   | <Enhance>-S           |
| Search Current Table                | PF6                   | <Enhance>-PF2         |

---

| Form Action          | Primary Key        | Alternate Key(s)   |
|----------------------|--------------------|--------------------|
| Top of Window        | <Enhance>-Up Arrow | <Enhance>-<Ctrl-K> |
| Top of List          | <Enhance>-Page Up  | <Enhance>-T        |
| Typeover Mode Toggle | <Enhance>-^        |                    |
| Undo                 | <Ctrl-X>           | <Enhance>-<Ctrl-X> |
| Up Arrow             | Up Arrow           | <Ctrl-K>           |

## WYSE85

### WYSE-85 (VT 220 Emulation) Keyboard Mapping

| Form Action                         | Primary Key           | Alternate Key(s)   |
|-------------------------------------|-----------------------|--------------------|
| Beginning of Field                  | <Enhance>-Left Arrow  | <Enhance>-<Ctrl-H> |
| Bottom of List                      | <Enhance>-Next Screen | <Enhance>-B        |
| Bottom of Window                    | <Enhance>-Down Arrow  | <Enhance>-<Ctrl-J> |
| Break                               | <Ctrl-C>              |                    |
| Delete Word                         | <Ctrl-W>              |                    |
| Delete Instance (Multi-line fields) | <Enhance>-Remove      |                    |
| Delete Row (Multi-row forms)        | Remove                |                    |
| Delete Previous Character           | Delete                |                    |
| Delete Character                    | <Ctrl-D>              |                    |
| Down Arrow                          | Down Arrow            | <Ctrl-J>           |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K> |
| Enhance                             | PF1                   |                    |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Delete   |
| Explain                             | PF2                   |                    |
| GETOUT                              | F20                   |                    |
| GETOUTALL                           | <Enhance>-F20         |                    |
| Go to Bottom Menu                   | PF3                   |                    |
| Help Menu                           | Help                  |                    |
| Insert Instance (Multi-line fields) | <Enhance>-Insert Here |                    |
| Insert Row (Multi-row forms)        | Insert Here           | <Enhance>-I        |

## Appendix B—Keyboard Actions

| Form Action                 | Primary Key           | Alternate Key(s)   |
|-----------------------------|-----------------------|--------------------|
| Left Arrow                  | Left Arrow            | <Ctrl-H>           |
| List Choices                | Select                | <Enhance>-L        |
| Move Window (in Quick-Form) | <Enhance>-Space       |                    |
| Next Field                  | Tab                   |                    |
| Next Word                   | <Ctrl-F>              |                    |
| Next Screen                 | Next Screen           | <Enhance>-N        |
| Previous Screen             | Prev Screen           | <Enhance>-P        |
| Previous Word               | <Ctrl-B>              |                    |
| PREVIOUS                    | PF4                   |                    |
| PROCEED                     | Do                    |                    |
| RETURN                      | Return                |                    |
| Right Arrow                 | Right Arrow           | <Ctrl-K>           |
| SAVE                        | F19                   | <Enhance>-S        |
| Search Current Table        | Find                  | <Enhance>-PF2      |
| Top of Window               | <Enhance>-Up Arrow    | <Enhance>-<Ctrl-U> |
| Top of List                 | <Enhance>-Prev Screen | <Enhance>-T        |
| Typeover Mode Toggle        | <Enhance>-^           |                    |
| Undo                        | <Ctrl-X>              | <Enhance>-X        |
| Up Arrow                    | Up Arrow              | <Ctrl-U>           |

## Xterm

### Xterm (X-Windows) Keyboard Mapping

| Form Action                  | Primary Key          | Alternate Key(s)                    |
|------------------------------|----------------------|-------------------------------------|
| Beginning of Field           | <Enhance>-Left Arrow | ^E Left Arrow                       |
| Bottom of List               | <Ctrl-V>             | <Enhance>-V<br><Enhance>-Page Down  |
| Bottom of Window             | <Enhance>-Down Arrow | <Enhance>-<Ctrl-J><br>^E Down Arrow |
| Break                        | <Ctrl-C>             |                                     |
| Delete Row (Multi-row forms) | F8                   |                                     |



| Form Action                         | Primary Key           | Alternate Key(s)                     |
|-------------------------------------|-----------------------|--------------------------------------|
| Delete Instance (Multi-line fields) | <Enhance>-F8          |                                      |
| Delete Word                         | <Ctrl-W>              | <Enhance>-W                          |
| Delete Character                    | Delete                | <Ctrl-D><br><Enhance>-D              |
| Delete Previous Character           | <--                   |                                      |
| Down Arrow                          | Down Arrow            | <Ctrl-J>                             |
| End of Field                        | <Enhance>-Right Arrow | <Enhance>-<Ctrl-K><br>^E Right Arrow |
| Enhance                             | F1                    | <Ctrl-G>                             |
| Erase Field Line                    | <Ctrl-L>              | <Enhance>-Delete<br><Enhance>-<--    |
| Explain                             | F2                    |                                      |
| GETOUT                              | F10                   |                                      |
| GETOUTALL                           | <Enhance>-F10         | <Enhance>-F4                         |
| Go to Bottom Menu                   | F3                    |                                      |
| Go to Top Menu                      | <Enhance>-F1          | <Enhance>-PF1                        |
| Help Menu                           | <Enhance>-F3          |                                      |
| Insert Instance (Multi-line fields) | <Enhance>-Insert      |                                      |
| Insert Row (Multi-row forms)        | Insert                | <Enhance>-I                          |
| Left Arrow                          | Left Arrow            |                                      |
| List Choices                        | F7                    | <Enhance>-L                          |
| Macro Definition                    | <Enhance>-K           |                                      |
| Next Field                          | Tab                   |                                      |
| Next Screen                         | Page Down             | <Enhance>-N                          |
| Next Word                           | <Ctrl-F>              | <Enhance>-F                          |
| Previous Field                      | <Enhance>-Tab         |                                      |
| Previous Screen                     | Page Up               | <Enhance>-P                          |
| Previous Word                       | <Ctrl-B>              | <Enhance>-B                          |
| PREVIOUS                            | F4                    |                                      |
| PROCEED                             | F5                    |                                      |
| RETURN                              | Return                |                                      |

| <b>Form Action</b>          | <b>Primary Key</b> | <b>Alternate Key(s)</b>           |
|-----------------------------|--------------------|-----------------------------------|
| Right Arrow                 | Right Arrow        | <Ctrl-K>                          |
| SAVE                        | F9                 |                                   |
| Search Current Table        | F6                 | <Enhance>-F2                      |
| Top of Window               | <Enhance>-Up Arrow | <Enhance>-<Ctrl-U><br>^E Up Arrow |
| Top of List                 | <Ctrl-T>           | <Enhance>-T<br><Enhance>-Page Up  |
| Typeover/Insert Mode Toggle | <Ctrl-^>           |                                   |
| Undo                        | <Ctrl-X>           | <Enhance>-X                       |
| Up Arrow                    | Up Arrow           | <Ctrl-U>                          |

## SQL Reserved Words

Do not use SQL reserved words for names of Open M/SQL objects, such as base tables, views, and fields.

|                    |           |            |          |
|--------------------|-----------|------------|----------|
| ALL                | DISTINCT  | MAX        | SQL      |
| AND                | DOUBLE    | MIN        | SQLCODE  |
| ANY                | END       | MODULE     | SQLERROR |
| AS                 | ESCAPE    | NOT        | SUM      |
| ASC                | EXEC      | NULL       | TABLE    |
| AUTHORIZA-<br>TION | EXISTS    | NUMERIC    | TO       |
| AVG                | FETCH     | OF         | UNION    |
| BEGIN              | FLOAT     | ON         | UNIQUE   |
| BETWEEN            | FOR       | OPEN       | UPDATE   |
| BY                 | FORTRAN   | OPTION     | USER     |
| CHAR               | FOUND     | OR         | VALUES   |
| CHARACTER          | FROM      | ORDER      | VIEW     |
| CHECK              | GO        | PASCAL     | WHENEVER |
| CLOSE              | GOTO      | ALI        | WHERE    |
| COBOL              | GRANT     | PRECISION  | WITH     |
| COMMIT             | GROUP     | PRIVILEGES | WORK     |
| CONTINUE           | HAVING    | PROCEDURE  |          |
| COUNT              | IN        | PUBLIC     |          |
| CREATE             | INDICATOR | REAL       |          |
| CURRENT            | INSERT    | ROLE       |          |
| CURSOR             | INT       | ROLLBACK   |          |
| DEC                | INTEGER   | SCHEMA     |          |
| DECIMAL            | INTO      | SECTION    |          |
| DECLARE            | IS        | SELECT     |          |
| DELETE             | LANGUAGE  | SET        |          |
| DESC               | LIKE      | SMALLINT   |          |



## Base Table Design Charts

The following charts are provided for your convenience in designing the base tables in your relational application. The charts provided are:

- n Base Table Definition
- n Base Table Fields
- n Field Definition by Data Type

### Base Table Definition

Base Table Name: \_\_\_\_\_ Routine

Name Prefix: \_\_\_\_\_

Description: \_\_\_\_\_ Global

Name: \_\_\_\_\_

| Structure?                   | Default          | Customized | Approximate # of Rows |  |  |  |
|------------------------------|------------------|------------|-----------------------|--|--|--|
| Row ID Fields                |                  |            |                       |  |  |  |
| Lookups                      | Fields to lookup |            | Fields to Display     |  |  |  |
| 1 Unqualified                | Leave blank      |            |                       |  |  |  |
| 2                            |                  |            |                       |  |  |  |
| 3                            |                  |            |                       |  |  |  |
| 4                            |                  |            |                       |  |  |  |
| 5                            |                  |            |                       |  |  |  |
| Characteristic Relationships |                  |            |                       |  |  |  |

|                           |                  |            |
|---------------------------|------------------|------------|
| Parent Base Table         | None or Name:    |            |
| Child Tables              |                  |            |
|                           |                  |            |
|                           |                  |            |
| Designative References    | Referenced Table | From Field |
|                           |                  |            |
|                           |                  |            |
| Validation Code           |                  |            |
| Triggers                  |                  |            |
| InsertPre ( )<br>Post ( ) |                  |            |
| UpdatePre ( )<br>Post ( ) |                  |            |
| DeletePre ( )<br>Post ( ) |                  |            |

Base Table Fields

Base Table

| Field         | Data Type | Req? | Unique? | Lookup? | If Computed, How? |
|---------------|-----------|------|---------|---------|-------------------|
| Row ID Fields |           |      |         |         |                   |
|               |           |      |         |         |                   |
|               |           |      |         |         |                   |



**Base Table Name:** \_\_\_\_\_

**Field Name:** \_\_\_\_\_ **Description:** \_\_\_\_\_

|                                            |                                                         |                           |  |
|--------------------------------------------|---------------------------------------------------------|---------------------------|--|
| Maximum Length                             | Unique? Yes No                                          | Number of Distinct Values |  |
| Field Protection:                          | Output Only<br>Add Add/Edit                             | Default Value             |  |
| Required? Yes No Maybe                     |                                                         |                           |  |
| If Maybe, M code to determine if required  |                                                         |                           |  |
| M/PACT Column Title                        |                                                         |                           |  |
| Multi-Line Field                           |                                                         |                           |  |
| Storage Mode                               | One Per Node ( )<br>All in One Node ( ) Delimiter: ____ |                           |  |
| Computed Field                             |                                                         |                           |  |
| Result Always = to the Computation? Yes No | If NO: only when these fields are updated               |                           |  |
|                                            |                                                         |                           |  |
|                                            |                                                         |                           |  |
| M code to compute result                   |                                                         |                           |  |
| NULL result IF                             |                                                         |                           |  |
|                                            |                                                         |                           |  |
|                                            |                                                         |                           |  |



|                                                                                              |                         |
|----------------------------------------------------------------------------------------------|-------------------------|
| Format:<br>1 10/18/934 18/10/93<br>2 18 Oct 935 Oct 18, 1993<br>3 18.10.936 October 18, 1993 | First Date<br>Last Date |
|----------------------------------------------------------------------------------------------|-------------------------|

## Field Definition

**Data Type:** MULTIPLE CHOICE

**Base Table Name:** \_\_\_\_\_

**Field Name:** \_\_\_\_\_ **Description:** \_\_\_\_\_

|                                            |                                                          |                           |
|--------------------------------------------|----------------------------------------------------------|---------------------------|
| Maximum Length                             | Unique? Yes No                                           | Number of Distinct Values |
| Field Protection:                          | Output Only<br>Add Add/Edit                              | Default Value             |
| Required? Yes No Maybe                     |                                                          |                           |
| If Maybe, M code to determine if required  |                                                          |                           |
| M/PACT Column Title                        |                                                          |                           |
| Multi-Line Field                           |                                                          |                           |
| Storage Mode                               | One Per Node ( )<br>All in One Node ( ) Delimiter: _____ |                           |
| Computed Field                             |                                                          |                           |
| Result Always = to the Computation? Yes No | If NO: only when these fields are updated                |                           |
|                                            |                                                          |                           |
|                                            |                                                          |                           |
| M code to compute result                   |                                                          |                           |

|                                       |  |  |
|---------------------------------------|--|--|
| NULL result IF                        |  |  |
|                                       |  |  |
|                                       |  |  |
| Multiple Choices (External  Internal) |  |  |
|                                       |  |  |
|                                       |  |  |
|                                       |  |  |

|                                                                                                                                                |                                                          |                           |
|------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|---------------------------|
| <div>Field Definition</div> <div>Data Type: NAME</div> <div>Base Table Name:</div> <div></div> <div>Field Name: Description:</div> <div></div> |                                                          |                           |
| Maximum Length                                                                                                                                 | Unique? Yes No                                           | Number of Distinct Values |
| Field Protection:                                                                                                                              | Output Only<br>Add Add/Edit                              | Default Value             |
| Required?Yes No Maybe                                                                                                                          |                                                          |                           |
| If Maybe, M code to determine if required                                                                                                      |                                                          |                           |
| M/PACT Column Title                                                                                                                            |                                                          |                           |
| Multi-Line Field                                                                                                                               |                                                          |                           |
| Storage Mode                                                                                                                                   | One Per Node ( )<br>All in One Node ( ) Delimiter: _____ |                           |

|                                                 |                                                        |  |
|-------------------------------------------------|--------------------------------------------------------|--|
| Computed Field                                  |                                                        |  |
| Result Always = to the Computation?    Yes   No | If NO: only when these fields are updated              |  |
|                                                 |                                                        |  |
|                                                 |                                                        |  |
| M code to compute result                        |                                                        |  |
| NULL result IF                                  |                                                        |  |
|                                                 |                                                        |  |
|                                                 |                                                        |  |
| Last, First M.                                  | Collation Function EXACT (default)<br>ALPHAUP<br>UPPER |  |

### Field Definition

Data Type:    NUMBER

Base Table Name: \_\_\_\_\_

Field Name: \_\_\_\_\_ Description: \_\_\_\_\_

|                                           |                                                                |                           |
|-------------------------------------------|----------------------------------------------------------------|---------------------------|
| Maximum Length                            | Unique?   Yes   No                                             | Number of Distinct Values |
| Field Protection:                         | Output Only<br>Add    Add/Edit                                 | Default Value             |
| Required? Yes   No   Maybe                |                                                                |                           |
| If Maybe, M code to determine if required |                                                                |                           |
| M/PACT Column Title                       |                                                                |                           |
| Multi-Line Field                          |                                                                |                           |
| Storage Mode                              | One Per Node   (   )<br>All in One Node (   ) Delimiter: _____ |                           |

|                                                                                                           |                                           |                                |
|-----------------------------------------------------------------------------------------------------------|-------------------------------------------|--------------------------------|
| Computed Field                                                                                            |                                           |                                |
| Result Always = to the Computation?<br>Yes No                                                             | If NO: only when these fields are updated |                                |
|                                                                                                           |                                           |                                |
| M code to compute result                                                                                  |                                           |                                |
| NULL result IF                                                                                            |                                           |                                |
|                                                                                                           |                                           |                                |
| Minimum                                                                                                   | Maximum                                   | Decimal Places                 |
| Leading Punctuation                                                                                       | Show Minus<br>-x x (x)                    | Used in Calculation?<br>Yes No |
| Format<br>1: 1234567.894: 1.234.567,89<br>2: 1234567,895: 1 234 567.89<br>3: 1,234,567.89 6: 1'234'567,89 |                                           |                                |

|                                                                                                                                     |                             |                           |
|-------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|---------------------------|
| <div>Field Definition</div> <div>Data Type: TEXT</div> <div>Base Table Name:</div> <div>Field Name: </div> <div>Description: </div> |                             |                           |
| Maximum Length                                                                                                                      | Unique? Yes No              | Number of Distinct Values |
| Field Protection:                                                                                                                   | Output Only<br>Add Add/Edit | Default Value             |
| Required?Yes No Maybe                                                                                                               |                             |                           |
| If Maybe, M code to determine if required                                                                                           |                             |                           |

|                                                        |                                                         |  |  |
|--------------------------------------------------------|---------------------------------------------------------|--|--|
| M/PACT Column Title                                    |                                                         |  |  |
| Multi-Line Field                                       |                                                         |  |  |
| Storage Mode                                           | One Per Node ( )<br>All in One Node ( ) Delimiter: ____ |  |  |
| Computed Field                                         |                                                         |  |  |
| Result Always = to the<br>Computation?    Yes   No     | If NO: only when these fields are updated               |  |  |
|                                                        |                                                         |  |  |
|                                                        |                                                         |  |  |
| M code to compute result                               |                                                         |  |  |
| NULL result IF                                         |                                                         |  |  |
|                                                        |                                                         |  |  |
|                                                        |                                                         |  |  |
| Collation Function EXACT (default)<br>ALPHAUP<br>UPPER |                                                         |  |  |

|                                                                                                                                                                 |                                |                           |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|---------------------------|
| <b>Field Definition</b><br><br><b>Data Type:    TIME</b><br><b>Base Table Name:</b><br><br>_____<br><b>Field Name:</b> _____ <b>Description:</b> _____<br>_____ |                                |                           |
| Maximum Length                                                                                                                                                  | Unique? Yes   No               | Number of Distinct Values |
| Field Protection:                                                                                                                                               | Output Only<br>Add    Add/Edit | Default Value             |
| Required? Yes   No   Maybe                                                                                                                                      |                                |                           |

|                                                                  |                                                         |  |  |
|------------------------------------------------------------------|---------------------------------------------------------|--|--|
| If Maybe, M code to determine if required                        |                                                         |  |  |
| M/PACT Column Title                                              |                                                         |  |  |
| Multi-Line Field                                                 |                                                         |  |  |
| Storage Mode                                                     | One Per Node ( )<br>All in One Node ( ) Delimiter: ____ |  |  |
| Computed Field                                                   |                                                         |  |  |
| Result Always = to the Computation?    Yes<br>No                 | If NO: only when these fields are updated               |  |  |
|                                                                  |                                                         |  |  |
| M code to compute result                                         |                                                         |  |  |
| NULL result IF                                                   |                                                         |  |  |
|                                                                  |                                                         |  |  |
| Format<br>1: 04:59PM<br>2: 16:59<br>3: 04:59:59PM<br>4: 16:59:59 | First Time<br>Last Time                                 |  |  |

|                                                                                                                                                  |                    |                           |
|--------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|---------------------------|
| <div>Field Definition</div> <div>Data Type:    TEXT</div> <div>Base Table Name:</div> <div>Field Name: _____</div> <div>Description: _____</div> |                    |                           |
| Maximum Length                                                                                                                                   | Unique?   Yes   No | Number of Distinct Values |

|                                                        |                                                         |               |  |
|--------------------------------------------------------|---------------------------------------------------------|---------------|--|
| Field Protection:                                      | Output Only<br>Add      Add/Edit                        | Default Value |  |
| Required? Yes No Maybe                                 |                                                         |               |  |
| If Maybe, M code to determine if required              |                                                         |               |  |
| M/PACT Column Title                                    |                                                         |               |  |
| Multi-Line Field                                       |                                                         |               |  |
| Storage Mode                                           | One Per Node ( )<br>All in One Node ( ) Delimiter: ____ |               |  |
| Computed Field                                         |                                                         |               |  |
| Result Always = to the Computation?      Yes    No     | If NO: only when these fields are updated               |               |  |
|                                                        |                                                         |               |  |
| M code to compute result                               |                                                         |               |  |
| NULL result IF                                         |                                                         |               |  |
|                                                        |                                                         |               |  |
| Collation Function EXACT (default)<br>ALPHAUP<br>UPPER |                                                         |               |  |

### Field Definition

Data Type: Yes/No

Base Table Name:

Field Name: \_\_\_\_\_

Description: \_\_\_\_\_

|                |                |                           |
|----------------|----------------|---------------------------|
| Maximum Length | Unique? Yes No | Number of Distinct Values |
|----------------|----------------|---------------------------|

## Appendix D—Base Table Design Charts

---

|                                                 |                                           |               |  |
|-------------------------------------------------|-------------------------------------------|---------------|--|
| Field Protection:                               | Output Only<br>Add    Add/Edit            | Default Value |  |
| Required? Yes   No   Maybe                      |                                           |               |  |
| If Maybe, M code to determine if required       |                                           |               |  |
| M/PACT Column Title                             |                                           |               |  |
| Computed Field                                  |                                           |               |  |
| Result Always = to the Computation?    Yes   No | If NO: only when these fields are updated |               |  |
|                                                 |                                           |               |  |
| M code to compute result                        |                                           |               |  |
| NULL result IF                                  |                                           |               |  |
|                                                 |                                           |               |  |



---

# Glossary of Terms

---

## Alternative Data Source Form

A non-database form that uses programmer-defined filing code to interact with a data structure other than the Data Dictionary.

## Application

A set of forms, reports, and other Open M with SQL objects and programs linked together by menus to form a structure that provides a working user interface to a database.

## Application Mode

The mode in which users interact with the Data Dictionary using forms and windows designed by the application programmer. In application mode, the user has no program development capabilities and never sees the M prompt. S/he enters and exits the application directly from the operating system prompt. Menus set up by the application designer govern navigation in application mode.

## Auxiliary Window

Any window defined for a form that is neither the form's master window nor its row selection window.

## Base Table

A collection of data represented in a simple 2-dimensional format consisting of one or more rows with one or more columns. Each row has at most one value for each column, and each row is unique, meaning that it differs by at least one column value from every other row. Also called a "relation" in relational database terminology.

### Characteristic Relationship

A user-defined join between base tables in which rows in a "child table" are existence-dependent on rows in a "parent table" in a many-to-one manner: a single parent row can have many child rows.

### Child Table

A base table that is existence-dependent on a parent table in a characteristic relationship. Rows in a child table must have contain a pointer to a row in the parent table.

### Code Generation

The process by which Open M with SQL generates executable routines from programmer specifications.

### Column

See Field.

### Compilation

The process in which program source code is translated by Open M with SQL's language compiler into executable M code (called object code) and stored in routines.

### Computed Field

A field whose value is derived from a calculation, often involving the values of other fields.

### Conversion Code

M code used by Open M with SQL to convert data values from internal storage formats to external display formats, and from external input formats to internal storage formats.

### Cross-Table Reference Form

A form that uses a Designative Display field to retrieve information from an outside base table, i.e. not the base table declared as the form's data source.

### Database Form

A form created by the Form Generator that retrieves its data from a base table in the Data Dictionary.

## Data Dictionary

The Open M with SQL component that describes the elements of the relational database, including both its conceptual content and the mapping of its logical data definition to physical data structures in the global database.

## Data Source

The Data Dictionary base table from which a database form retrieves its data. A form may have only one data source. Non-database forms do not have any data source.

## Designative Display Field

A form-only field created from a Designated Reference field (in the associated base table) that retrieves information from a specified column (field) in the designated table. Designative Display fields may appear on a form.

## Designative Reference

A user-defined join between two base tables in which one field of the designating table contains Row IDs of rows in the referenced table. In relational database terminology, the designating table has a "foreign key" on the referenced table.

## Embedded SQL

SQL statements that are directly embedded within M routines. These statements are prefixed by the M binding &sql.

## Export/Import

Window-based facility that allows you to port Open M with SQL base tables, views, forms, reports, queries, menus, routines, and globals between different directories and different computers.

## External Value

The value of a field as revealed by Open M with SQL for all external display, or the value entered by the user in a form or in an SQL statement. A field may have both an internal and an external value.

## Field

A named unit of data in a base table row, usually representing a real world entity, such as a name, social security number, or date of birth for that row. Also called "column" or "attribute".

### Field Caption

The descriptive text attached to a field on a window.

### Foreign Key

See Designative Reference

### Form

One or more related windows referencing the same base table and their corresponding field definitions. Used to add, retrieve, delete, and modify data from the associated base table.

### Form Generator

The Open M with SQL application generator product, which allows you to design highly sophisticated forms and windows that interact with the relational Data Dictionary to add, update, retrieve, and delete data.

### Form-Only Field

A field that is created and exists only at the form level, independent of any Data Dictionary association. Fields of this type may appear on a form but cannot file data to the Data Dictionary. All fields on a non-database form are form-only.

### Form-Only Form

A non-database form that does not file data.

### Full Screen Editor

An Open M with SQL utility that allows you to create, edit, and view macro source routines, intermediate code routines, and include files.

### Global

A disk-based data storage unit specified by the M programming language standard. Also called "global variables", these are commonly implemented using balanced-tree technology.

### Global Database

The underlying logical and physical data storage structure of Open M with SQL, in which all data is stored in a system of multiply-subscripted arrays called "globals". Relational tables are mapped to the global database through the Data Dictionary.

## Implicit Join

A user-defined join between related tables defined in the Data Dictionary. In Open M with SQL, implicit joins can be characteristic relationships or designative references. Implicit joins facilitate data access by simplifying WHERE clauses.

## Include File

Files containing definitions that can be used in the preprocessor phase of compilation to expand macro source routines and determine whether optional lines of code should be included. They can also be used to include a common block of code in several routines, saving the overhead of calls to a common subroutine.

## Index Map

A mapping from one or more field values into the Row ID of each row in a table that speeds up access to rows when these values are known. It allows rapid retrieval of rows sorted by one or more field values.

## Integrity Constraints

User-defined constraints on data insert, update, and delete operations that ensure the accuracy and completeness of the application and the underlying database.

## Intermediate Source Code

The standard 3GL M source code available in all M implementations. Intermediate code is produced from macro source by the Open M with SQL compiler. At this level, all preprocessor syntax, including embedded SQL, has been resolved, and the routine contains only pure M source code. It is possible to write M routines at this level, but without the benefit of embedded SQL or other preprocessor syntax, such as macros.

## Internal Value

The value of a field as stored internally by Open M with SQL. A field may have both an internal and an external value.

## Join

A request for information from a base table (via a query, form, or report) in which data must be retrieved from more than one table, necessitating a link between base tables.

### Learn-As-You-Go

The ability for a form that makes a cross reference via Designative Display field to a another form owned by a different base table to add new rows to the designated table at runtime.

### Lookup Specifications

Lookup specifications define how a single-row form searches and retrieves rows from the database.

### M/PACT

The Open M with SQL report writer product and environment. Allows you to extract very specific data from the relational Data Dictionary and output that data in reports of row and column format with the benefit of many format enhancement features.

### Macro Source Code

The highest, most flexible and permissive level of code at which routines can be written. Macro source code permits the definition of macros and embedded SQL statements using a combination of ANSI-Standard M syntax, special macro pre-processor commands, and ANSI-Standard SQL.

### Map

In Open M with SQL, a map describes the relationship of the logical structure of a table to the physical structure of the global database.

### Master Map

The single master map for each base table defines the global structure for all of the data fields of the table. In M terminology, the master map defines the "upright file."

### Master Window

The introductory window of any form. Every form must unconditionally have a master window.

### Menu

A structure used for tying together and governing navigation through the various components of an application. Menus consist of one or more selectable options and may display those options vertically (top to bottom) or horizontally (left to right).

**Multi-Row Form**

A form that presents multiple rows of data from its associated base table simultaneously. The windows of a multi-row form display their set of fields in repeating units, each unit corresponding to one database row.

**M Language**

An ANSI-Standard procedural programming language, which was formerly called MUMPS.

**National Language Independence**

Open M with SQL environment customization feature that allows you to convert system-generated runtime messages and menus to the language best suited to the users at a particular installation site as well as tailor the display formats of numbers and dates according to national conventions.

**Non-Database Form**

A form that is not associated with a base table in the Data Dictionary, i.e., has no data source.

**Object Code**

The lowest level of code produced by the Open M with SQL compiler. This is the code that is actually interpreted and executed. Routines cannot be written at the object code level.

**OPENM.DAT**

The primary volume of your M database.

**Parent Reference Field**

Field automatically created to define the Row ID in the dependent child table when a characteristic link is defined. The Parent Reference Field acts like a designative reference from the child table to the parent table and has the same name as the parent table.

**Primary Key**

A field or combination of fields used to uniquely identify each row of a base table. In Open M with SQL, the function of the primary key is performed by the Row ID.

### Privilege

The authority of a user to perform an action on an object. The owner of an object grants and revokes privileges on that object.

### Programmer Mode

The mode in which all program development activity takes place. In programmer mode, you initiate all activity from the M prompt, and the M prompt reappears at the conclusion of every program you run. Programmer mode encompasses the M environment and all programs that can be called from it, including the Open M with SQL development environment as well as Open M with SQL runtime programs. In programmer mode, you create applications that users subsequently run in application mode.

### Prompt

System-generated signal requesting some user response.

### Query

An SQL language construct that enables the interrogation and manipulation of the database in a relationally complete manner. In Open M with SQL, queries can be embedded directly within M code, or may be written interactively using the Query Editor or the Query Generator (for SELECT-statement queries only).

### Referential Integrity

Referential integrity constraints ensure that database insert, update, and delete operations that apply to tables linked by implicit joins do not compromise the accuracy and completeness of the database.

### Relation

Indicates a link between base tables. See Characteristic Relationship and Designative Reference.

### Relational Database

A collection of related tables. See Base Table.

### Report

A program defined using the M/PACT report writer that retrieves and displays data from the relational Data Dictionary.



**Required Field**

A field in a table that must contain a valid non-null value before the row can be filed.

**Row**

A group of related field values that describes an entity in the domain of the table. For example, in a Customers table, a row describes a single customer.

Also called a "record" in traditional data processing terminology, or a "," in relational database terminology.

**Row ID**

In Open M with SQL, the field in a row whose value uniquely identifies that row in the table. In relational terminology, the Row ID functions as the primary key.

**Row Selection Window**

A window that is automatically generated by Open M with SQL for single-row forms based on the lookup specifications defined by the programmer. The row selection window is the first window to appear when the form is called and is used to select the database row to be retrieved. Multi-row forms do not have row selection windows.

**Single-Row Form**

A form that presents data from its associated base table one row at a time.

**SQL**

Structured Query Language. SQL is the ANSI-standard fourth generation programming language designed specifically for accessing and maintaining relational data bases.

**Stand-Alone Caption**

Descriptive text that appears on a window but is not attached directly to a field. Often used in the capacity of a window header.

**Trigger**

User-defined SQL or M code that executes automatically in response to 1.) insert, update, or delete operations performed on base tables in order to enforce referential integrity and 2.) data entry in an M/WINDOWS form in order to enforce application integrity.

### Validation Code

M code that validates row and field values. Base Table Validation Code specifies a condition that must be satisfied among all the fields in a row before the row can be filed. Field Validation code specifies constraints on individual field values.

### View

A virtual table created using the fields from a base table or set of base tables linked by implicit joins.

### Virtual Field

A field that does not correspond directly to a single stored value possibly composed of several stored values. For example, the Row ID field for a child table is sometimes composed of two stored values: the Row ID of the parent and a sub-script corresponding to a particular child row.

### Virtual Table

A table not directly represented in physical storage. Views and the results of queries are virtual tables.

### Window

A set of fields and captions displayed together on the screen.

---

# Index

---

## A

- Access Level 18-16
  - default physical structure 16-16
  - deleting specifications 18-27
  - special access code 18-29
  - specifying how to reach 18-20
- Access Path Specifications 18-9
  - access type 18-21
  - data access expression 18-17
  - default physical structure 16-16, 16-18
  - defining in ^mxdd global 20-18
  - defining standard 18-14
  - examples 18-26
  - master window 18-15
  - special access code 18-29
  - testing 17-14
  - value expression 18-22
- Access Type 18-16, 18-21
  - default 18-16
  - full row location 18-22
  - Global 18-21
  - Other 18-10, 18-19, 18-21
  - Piece 18-21
  - Sub 18-19, 18-21, 18-32
- ^ACCT global example 17-3, 17-22, 18-2
  - access path specifications 18-14
  - SET command 18-5
  - syntax 17-22
- Add protection 6-47, 6-48
- Advanced Base Table Options window 12-4
- All in One Node storage mode 6-38, 6-41
- ALPHAUP 6-29, 8-16, 8-22
- \$\$ALPHAUP function 18-24
  - index maps 18-23
  - names of tables 20-3
- %ALTER privilege
  - base table 12-2
  - copying a view 14-28
  - deleting a base table 5-23
  - deleting a view 14-24
  - editing a base table 5-21
  - editing a view 14-23
  - editing/deleting a field 6-59
  - privilege node 20-30
- ALTER VIEW statement 14-20
- Alternative Data Source Form Glossary-1
- AND clause
  - example 2-21, 8-24
- Application development
  - existing applications 1-16
  - open systems architecture 1-5
  - productivity 1-15
  - relational applications 1-6
  - via relational server 1-5
- Application mode Glossary-1
- Applications Glossary-1
  - COSTAR 18-14
  - data retrieval tools 1-5
  - default field value 6-47
  - defining the data structure 1-8, 4-3

- designing 4-1
- existing 17-2
- importing data definitions 20-1
- mapping functional specifications 1-7, 4-2
- relational view data 17-5
- sample application 4-4

Arrow syntax 2-21, 7-7  
See also Extended arrow syntax

Auxiliary Window Glossary-1

## B

Base table 2-3, 5-1, Glossary-1

- advanced options 5-10
- approximate number of rows 5-8
- characteristic relationships 2-19, 5-11
- compiling 5-17, 5-23
- creating under customized physical structure 17-9
- data on disk 2-4
- defining in ^mxdd import global 20-5
- defining lookups 8-3
- definition 5-1
- deleting definition 5-23
- description field 5-6
- designative reference 2-18
- editing definition 5-21
- field length check 5-10
- first normal form 2-6
- help messages 5-10, 12-1
- help topics 12-4
- identifying in ^ACCT global example 17-20, 17-23
- identifying in ^FLAVORS global example 17-29
- in view 14-14
- irn 5-10, 12-4, 16-12
- listing all tables 15-4
- listing fields report 15-5
- listing multiple 15-7
- message translations 12-10
- name in master map 16-13
- naming conventions 5-4
- override lock reference 5-10
- overview 1-9
- owner 5-10, 5-23, 12-2, 12-4

- relationships report 15-8
- report 15-4
- routine name prefix 5-17
- routes 5-17
- row insert and delete messages 5-10, 12-5, 12-7
- sample application 4-5
- saving definition 3-13
- top-level nodes 20-6
- triggers 11-2
- validation code 1-16

Base Table Definition window 5-9

Base table fields 1-11, 6-1, Glossary-3

- add/edit protection 6-48
- attributes 1-12
- checklist for defining 6-3
- column number 6-47, 6-60
- column number listing 15-6
- computed 6-7, 6-42, Glossary-2
- conversion and validation order 10-10
- conversion code 10-4
- data entry 6-2
- data type 6-7, 6-9
- %data variable 11-8
- default value 6-47, 6-50
- defining in ^mxdd import global 20-7
- defining location 18-45
- deleting data 6-59
- description field 6-5
- %edit variable 11-8
- editing and deleting 6-59
- error messages 13-2
- external value 10-2, Glossary-3
- external-to-internal conversion code 10-4
- field name 6-5
- field name for map definition 18-4
- full row reference value 18-47, 18-52
- global storage 16-12
- global structure 16-19
- help messages 13-2
- ID number 6-47
- identifying in ^ACCT global example 17-24
- identifying in ^FLAVORS global example 17-29, 17-31
- identifying in existing M database 17-7

- in offset of Full Row Reference 18-49
  - in views 14-11, 14-14, 14-15, 14-27
  - index maps 9-4
  - internal value 10-2, Glossary-5
  - internal-to-external conversion code 10-5
  - length checking 5-10, 12-5
  - level below Full Row Reference 18-49
  - listing report 15-5
  - lookup 1-9, 6-53
  - maximum length 6-7, 12-5
  - maximum number 6-2
  - multi-line 6-7, 6-37
  - naming conventions 6-5
  - node 15-15
  - node below Full Row Reference 18-53
  - nodes 20-7
  - NULL values 17-7, 17-26
  - number of distinct values 6-7, 6-35
  - output only protection 6-48
  - piece of a piece of node value 18-49
  - pieces of node value 18-48
  - printing a list of 15-5
  - protection 6-47, 6-48
  - required 6-47, 6-49, Glossary-9
  - sample application 4-7
  - saving definition 6-56
  - triggers 11-8
  - unique fields 6-7, 6-34
  - update features 6-7, 6-46
  - validation code 1-16, 10-7, 12-5
- BName subscript 20-3
- C**
- \$C  
See \$CHAR function
- C programming language  
applications 1-5
- Caption Glossary-4  
stand-alone Glossary-9
- Cartesian product operation 2-10
- Case transformation  
See Collation sequence
- Case transformation functions  
See ALPHAUP, UPPER, EXACT
- \$CHAR function 6-41
- Characteristic relationship 2-19, 5-11, 7-5,  
Glossary-2  
See also Parent table, Child table  
child-to-parent relationship 2-22  
defining 5-11  
description 1-10  
example 7-6  
identifying in ^ACCT global example 17-24  
identifying in ^FLAVORS global example 17-30  
identifying in existing M database 17-6  
implicit join syntax 2-22  
inner join 7-5  
listing all 15-8  
lookups 8-13  
parent-to-child relationship 2-22  
pre-delete trigger 5-21  
relationships report 15-8  
Row ID field 16-5  
view 14-2, 14-8
- Checklist  
base table definition 5-2  
field definition 6-3  
map definition 18-3  
relational definition 17-4
- Child table 2-19, 5-11, 7-5, Glossary-2  
arrow syntax 2-21  
cannot change status 5-21  
childsub field 16-6, 16-8  
creating 5-11  
defining from parent table 17-9  
defining lookups for 8-13  
designative reference to 7-4  
parent reference field 5-21, 16-6  
Row ID field 16-5, 17-11  
row ID field example 16-6
- childsub field 16-6, 16-8, 17-27  
in master map 16-13  
modifying 17-32
- Code generation Glossary-2
- Collation sequence  
ALPHAUP 6-29  
EXACT 6-28  
name data type 6-19

- ORDER BY clause 6-31
  - string comparisons 6-31
  - text data type 6-24
  - UPPER 6-30
  - Column number
    - determining field's 6-47, 6-60
    - fields in %data and %edit 11-8
    - listing fields by 15-6
    - parent reference field 6-47, 16-6
    - row ID field 6-47
  - Columns 2-2
    - See Base table fields
  - Comments
    - display sequence 12-13
    - for base table 5-10, 12-12
    - ^mcomment global 12-11
    - searching 12-13
    - views 14-22
  - Comparison operators
    - collation sequence 6-31
  - Compiling Glossary-2
    - base table 5-17
    - base table after editing 5-23
    - create default structure 5-16
    - not required for table name change 5-6
    - populate index maps 9-8
    - related objects 5-9, 5-20
    - specify routine name 5-10
  - Computed field 6-7, 6-42, Glossary-2
    - always equal to computation 6-42, 16-20, 17-8
    - benefits 6-42
    - maximum number of fields referenced 6-42
    - multi-line 6-37
    - not always equal to computation 6-43
    - types 6-42
    - variables in calculation 6-42
  - Computed Field window 6-44
  - Conditional map 16-17
    - creating 17-17
    - defining 18-58
    - example 18-60
  - Conversion code 6-7, Glossary-2
    - automatically generated 8-17, 10-3
    - date fields 6-14
    - external-to-internal 10-4
    - fields 10-4
    - internal-to-external 10-5, 18-56
    - overview 10-2
    - variables allowed 10-9
  - Conversion/Validation Code window 10-12
  - COSTAR applications
    - data stored in multiple globals 18-14
  - CREATE VIEW statement 14-20
  - Cross-Table Reference Form Glossary-2
  - Customized physical structure 18-1
    - ^ACCT global example 17-22
    - beginning with 18-6
    - computed fields 6-42, 6-43
    - conditional map 16-17
    - creating map definition 18-1
    - creating preliminary Master Map 17-13
    - ^FLAVORS global example 17-29
    - index maps 9-7, 17-16
    - override filing code 16-17
    - row ID field 2-3
    - ^SAMPLE global example 17-19
    - testing mapping 17-14, 17-17
    - when to use 1-14
- ## D
- Data
    - accessing 1-9
    - conversion 10-2
    - converting from FileMan data 19-1
    - maximum length 6-33, 12-5
    - not deleted when field definition deleted 6-59
    - sorting by field values 9-4
    - validation 10-2, 12-5
  - Data access
    - efficiency 7-8
    - loop 18-30
    - single value 18-29, 18-33
  - Data Access Expression 18-16, 18-17
    - access level 1 18-11
    - customized physical structure 18-18
    - default 18-17
    - default physical structure 16-18

- defining override 18-19
- forms 18-19
- global reference 18-19
- names 18-17
- NULL 18-19
- piece 18-19
- Data Access Variables
  - defining 18-16
  - defining in ^mxdd global 20-21
  - M Code window 18-40
  - names 18-39
  - specifying 18-39
  - where used 18-39
- Data conversion
  - See Conversion code
- Data Dictionary Glossary-3
  - accessing 3-2, 3-5
  - accessing data from 2-9
  - benefits 1-15
  - defining base table fields 1-11, 6-1
  - defining base tables 1-9, 5-1
  - defining data structures 1-8, 4-3
  - defining views 1-12, 14-1
  - exiting 3-13
  - getting started 3-1
  - importing data definitions 20-1
  - main menu 3-4
  - maximum values A-1
  - overview 1-2
  - reports 15-1
  - saving definitions 3-13
- Data entry 16-2
  - field conversion and validation sequence 10-10
  - field length validation 12-5
- Data entry mode 3-8
- Data entry window 3-8
- \$DATA function 18-29, 18-30, 18-33
  - conditional trigger item 11-18
  - single loop access validation 18-31
- Data source Glossary-3
- Data type 6-7
  - common errors 6-10
  - conversion and validation code 10-3
  - defining 6-9
  - existing M database 6-10
- Data validation
  - See Validation code
- %data variable 10-9
  - column number 11-8
  - designative reference value 11-17
  - field subscripts 6-47
  - filing code 18-55
  - in triggers 11-8
  - trigger conditional execution 11-18
  - trigger type and value 11-9
- Database 1-7, 2-1
  - define base table fields 6-1
  - define base tables 5-1
  - for non-ISM systems 2-8
  - relational 2-2
- Database Form Glossary-2
- %DATE 10-5
- Date data type 6-11
  - additional validation code example 10-8
  - conversion code 6-14, 10-3, 10-4, 10-6
  - conversion code example 10-5, 10-6
  - conversion/validation code 10-12
  - data type errors 6-10
  - default value 6-51
  - defaults 6-9
  - delimiter 6-12
  - existing data 6-10
  - imported fields 6-54
  - internal storage format 6-13
  - space requirements 6-2
  - t variable 6-12
  - validation code 10-3, 10-7
  - year display 6-12
- ^DD global
  - FileMan interface 19-8
- Default physical structure 16-1
  - access path specifications 16-18
  - data access expression for Level 1 16-18
  - full row reference 16-19
  - generating 5-15
  - index maps 9-3, 9-5, 16-22
  - index maps for imported data 20-23
  - index maps for lookups 8-3
  - index maps for unique fields 6-34

- map data specifications 16-19
  - multi-line field storage mode 6-37, 16-14
  - overview 16-2
  - row ID field 2-3, 5-16
  - row ID specifications 16-20
  - sample application 4-4
  - when to use 1-14
  - window 16-10
- DELETE filing code field 18-55, 18-56
- DELETE privilege
- for base table 12-2
- DELETE trigger 11-9
- field value 11-8
  - pre-filing 11-4, 11-5, 11-9, 11-17
  - pre-filing example 11-19
- Designated table 2-18
- See Referenced Table
- Designating table 7-2
- Designative display fields 8-10, Glossary-3
- Designative reference 2-18, 7-2, Glossary-3
- arrow syntax 2-21
  - conversion code 10-4, 10-6
  - conversion/validation code 10-3
  - data type error 6-10
  - defaults 6-9
  - defining 6-14
  - designative display field 1-11
  - example 7-2
  - existing data 6-14
  - foreign key 7-2
  - identifying in existing M database 17-7
  - implicit join syntax in queries 2-21
  - listing all 15-8
  - number of distinct values 6-36
  - outer join 7-3
  - overview 1-10
  - privileges required 12-2
  - relationships report 15-8
  - to child table 5-14
  - validation code 10-7
  - view 14-2, 14-8
- ^DIC global
- FileMan interface 19-8
- %DN variable 10-5
- DROP VIEW statement 14-20
- %DS variable 10-5
- DSM implementation 2-8, 16-26
- DTM implementation 16-26
- ## E
- \$E
- See \$EXTRACT function
- %edit variable 10-9
- array 11-21
  - column number 11-8
  - field subscripts 6-47
  - filing code 18-55
  - in triggers 11-8
  - trigger conditional execution 11-18
  - trigger type and value 11-9
  - trigger update 11-5
- Embedded SQL Glossary-3
- Equijoins
- See Inner joins
- Error and Help Messages master window
- 13-2, 13-5
- Error messages
- defining 13-4
  - field 6-8
  - for fields 13-1, 13-2
  - languages supported 13-4
  - translations 13-4
- EXACT 6-28, 8-16, 8-22
- %EXACT SQL extension function 8-22
- Existing M database
- ^ACCT global example 17-22
  - ^FLAVORS global example 17-29
  - indices 17-16
- EXPLAIN key 13-2
- Export/Import facility Glossary-3
- Extended arrow syntax
- example 7-7
  - simplifies queries 7-7
  - view join specification 14-9, 20-27
- Extended global reference 18-11



External value Glossary-3  
\$EXTRACT function 18-22  
    defining Row ID location 17-8  
    FileMan fields 19-7

## F

FErrTrans subscript 20-13  
FHelpTrans subscript 20-14  
Field Definition main window 6-6  
Field Name 6-5  
    during map definition 18-4  
Field protection 6-48  
Fields  
    form-only Glossary-4  
    See also Base table fields  
Fields on View Definition window 14-15  
Fields to Include in View window 14-26  
File 6-54  
FileMan interface 19-1  
    accessing 19-8  
    conversion error log 19-27  
    conversion logic 19-2  
    creating the link 19-10  
    cross reference report 19-25  
    data format 6-54  
    database normalization 19-2  
    date/time format 6-54  
    deleting the link 19-20  
    extending the link 19-15  
    field mapping 19-7  
    field specifications 19-6  
    file specifications 19-5  
    generating views 19-23  
    identifiers 19-3  
    indices 19-7  
    limitations 19-30  
    link maintenance facility 19-13  
    overview 19-2  
    PDP machines 19-29  
    purge conversion error log 19-28  
    questions and answers 19-32  
    ScreenMan applications 19-7  
    updating the link 19-18  
    variable pointer fields 19-6

Filing code 17-19  
    See also Override filing code  
    %data variable 18-55  
    default 18-55  
    %edit variable 18-55  
    filing arrays 18-55  
    in base table triggers 11-4  
    multi-line fields 18-55  
    not generated 18-46  
    override 18-51  
First normal form 2-6  
^FLAVORS global example 17-3, 17-29, 18-2  
    access path specifications 18-14  
    syntax 17-29  
FName subscript 20-3  
Foreign key 2-18, 7-2, Glossary-4  
    See Designative reference  
Foreign languages  
    See Translations  
Form Generator 1-4, Glossary-4  
    column header 6-5  
    computed field 6-42  
    defining lookups 8-3  
    designative display field Glossary-3  
    field caption 6-5  
    field maximum length 6-7  
    field protection 6-48  
    forms affected by field definition 6-2  
    Keyboard actions B-1  
    multi-line fields 6-37  
    row delete message 12-5  
    row ID field 6-51  
    row insert message 12-5  
    triggers 11-2  
    views 14-2  
Form-only fields Glossary-4  
Forms Glossary-4  
    alternative data source Glossary-1  
    cross-table reference Glossary-2  
    database Glossary-2  
    designative display field Glossary-3  
    form-only Glossary-4  
    multi-row 3-5, Glossary-7  
    navigating 3-9  
    non-database Glossary-7

- single-row 3-5, Glossary-9
- FROM clause 2-21
  - example 2-21
  - query options 8-20
- Full Row Reference 16-12, 16-17, 18-10
  - \$DATA function 18-33
  - default 16-16, 18-41
  - default physical structure 16-19
  - example 17-26
  - listing 15-15
  - %row variable 16-19, 18-41
  - specifying override 18-41
  - validity testing 18-29
- Full Screen Editor Glossary-4
  - horizontal options menu 3-8
  - status line 3-7

## G

- GETOUT key 3-11
- GETOUTALL key 3-11
- Global 1-3, Glossary-4
  - bracket syntax 16-26, 18-11
  - COSTAR applications 18-14
  - database Glossary-4
  - definition of 17-5
  - environment syntax 18-11
  - extended global reference 18-11
  - multi-line fields 16-13
  - ^mxdd 20-2
  - naming conventions 18-11
  - reference 17-5
  - report on mapped globals 15-13
- Global name 18-9
  - access level 1 18-11
  - asterisk 18-9
  - customized physical structure 18-8
  - default value 5-13
  - defining in default physical structure 16-25
  - editing 5-15, 5-21
  - editing default 5-15
  - specifying 18-11, 18-12
- Global structure 1-3
  - base table fields 16-19
  - defined in Master Map 16-12

- indexes in existing M database 17-16
- multi-line field 16-14
- new format 16-27
- old format 16-27

Go to Bottom Menu key 3-10

GROUP BY clause 8-20

## H

\$H

See \$HOROLOGY function

HAVING clause 8-20

Help facility 3-11

Help messages

- array reference 13-8
- defining 13-4
- field 6-8
- for fields 13-1
- help key 13-2
- long help 13-4
- override help 13-3
- short help message 13-5
- translations 13-4

Horizontal options menu 3-8

\$HOROLOGY function 6-13, 10-3, 17-23

- computed field calculations 6-42

## I

Implicit joins 2-18, 7-1, Glossary-5

- arrow syntax 2-21
- as characteristic relationships 2-19, 5-11
- as designative references 2-18
- benefits 7-7
- extended arrow syntax 7-7
- integrity constraints 2-23
- referential integrity 7-8
- report 15-8
- syntax for child-to-parent references 2-22
- syntax for designative reference 2-21
- syntax for parent-to-child references 2-22
- view 14-9, 14-14

Importing data definitions 20-2

- ^mxdd global 20-2

- Include file Glossary-5
- Index Maps 17-6, Glossary-5
  - accessing definition 16-10
  - additional fields to index 9-5
  - advantages and disadvantages 9-2
  - creating customized 17-16, 18-5
  - customized physical structure 9-7, 18-5
  - default physical structure 9-5, 16-2
  - defining as conditional 16-17, 18-58
  - defining default in ^mxdd global 20-23
  - deleting a field 6-59
  - examining default 16-22
  - existing M database 17-16
  - fields to index 9-4, 16-22
  - fields with NULL value 9-3
  - from FileMan indices 19-7
  - functions 18-23
  - in ^mxdd import global 20-23
  - lookups 8-3
  - name data type fields 16-24
  - naming conventions 18-6
  - new format 16-27
  - old format 16-27
  - Row ID field 9-3
  - structure 9-3
  - unique fields 6-34
- indn 16-27
- Inner joins 2-14
  - characteristic relationship 7-5
- Inquiry mode 3-8
- INSERT filing code field 18-56
- INSERT privilege
  - for base table 12-2
- INSERT statement
  - multi-line field line count 6-41
- INSERT trigger
  - field value 11-8
  - pre-filing 11-4, 11-5, 11-9, 11-17
  - pre-filing example 11-17
- Integrity
  - constraints 7-8, Glossary-5
  - referential 1-16, Glossary-8
- Intermediate source code Glossary-5
- Internal value Glossary-5
- irn 5-10, 12-4, 16-12, 16-27
- %IS output device 15-3
- ISM implementation 16-26
- J**
- \$j 12-5
- Join Specifications 14-8
  - examples 14-10
  - for view 14-7
  - report listing 15-9
- Joins 2-14, 7-1, Glossary-5
  - equijoin 2-14
  - implicit joins 2-18, 7-1
  - inner joins 2-14
  - one-way outer joins 2-16, 7-3
- K**
- Keyboard actions 3-9
  - frequently used 3-10
  - on-line help 3-11
- Keyboard mapping B-1
- L**
- Languages
  - See Translations
- Learn-As-You-Go Glossary-6
- %linenum variable 10-9
- Local array storage 18-12
  - defining 18-19
  - global name 18-9
  - help message text 13-3, 13-8
  - multi-line fields 18-55
  - naming conventions 18-19
  - special access code 18-14
- %local variable
  - current access level 18-35
- Lookup Specifications window 8-8
- Lookups 8-1, Glossary-6
  - child table 8-13
  - collation sequence 6-32
  - compound lookup query 8-6
  - computed field 6-43

- customize query 8-19
- default query 8-18
- defining 8-3, 8-7
- deleting a field 6-59
- designative display fields 8-10
- enabling lookup queries 8-6
- exact matching 8-15
- example 8-2
- exclude field 6-53
- fields to display 8-9
- fields to lookup on 1-9, 6-53, 8-1
- index map for 8-3, 9-4, 16-22
- lookup display fields 8-4
- lookup fields 8-3
- lookup query 8-2
- matching 8-14
- override query 8-18
- partial matching 8-15
- precedence of queries 8-6
- queries 8-8
- query types 8-5
- Search Current Table key 8-4
- single field lookup query 8-5
- SQL extensions 8-21
- suppress lookup box 8-19
- toggle between default and override query 8-19
- user input variables 8-23

## M

### M database

- designing a relational definition 17-5
- existing 1-8, 1-16, 17-2, 20-1
- existing and field data types 6-10
- for non-ISM systems 2-8
- mapping links to relational view 1-13
- new 17-2
- physical structure 1-9
- referential integrity 1-16
- time stamp format 6-54
- viewing as relational database 17-1

### M directory

- as schema 2-8

### M language 1-3, Glossary-7

- SET command 1-3

### M/PACT Glossary-6

- column header 6-5
- column headers 6-5, 6-54
- column titles defined in view 14-12, 14-15, 14-27
- computed field 6-42
- data from multiple tables 2-5
- field maximum length 6-7
- index map for 9-4
- multi-line fields 6-37
- reports affected by field definition 6-2
- time display format 6-26
- view as data source 2-5, 14-3
- view as source for 14-2

### Macro source code Glossary-6

### Map Data Specifications 18-10

- default physical structure 16-19
- defining 18-45
- defining in ^mxdd global 20-23
- entering filing code 18-51
- examples 18-52
- multi-line fields 16-19
- node field 18-45
- piece and delimiter 18-46
- retrieval M code 18-50
- standard 18-45

### Map Definition master window 16-15, 18-8

- entering 18-6

### Mapping Glossary-6

- access path specifications for default structure 16-18
- Advanced Options window 18-56
- computed fields 6-42, 6-43
- creating customized map definition 18-1
- defining in ^mxdd global 20-17
- deleting default mapping 18-16
- field names 18-4
- map definition checklist 18-3
- map name 18-9
- naming conventions 18-6
- new format 16-27
- old format 16-27
- overview 1-13
- report 15-15
- report by global 15-13
- See also Default physical structure, Customized physical structure

- See also Global
  - SET command to insert row 18-5
  - testing customized 17-14, 17-17
  - Master Map 1-13, Glossary-6
    - accessing definition 16-10
    - base table name in 16-13
    - creating preliminary for customized
      - physical structure 17-13
    - customized physical structure 18-6
    - default physical structure 16-2
    - defining map as 18-9
    - editing default 18-7
    - examining default 16-12
    - irn 5-10, 16-12
    - naming conventions 18-6, 18-9
    - new format 16-13, 16-27
    - old format 5-10, 16-12, 16-27
    - testing 17-14
  - ^mcomment global 12-11
  - ^mdata global 5-13, 5-15, 16-2, 16-10, 16-12, 16-16, 16-25
  - Menu Glossary-6
  - Message line on window 3-9
  - Messages For Form Generator window 12-8
  - MIIS date and time formats 6-54
  - MName subscript 20-3
  - %msg variable 10-9, 10-14
    - base table validation 10-16
    - in conversion code 10-6
    - overrides error message 13-2
  - MSM implementation 16-26
  - ^%msql command 3-2
  - mt# routine name prefix 5-17
  - Multi-line field 2-23, 6-7, 6-37
    - All in One Node global storage 6-38
    - computed 6-37
    - default physical structure 16-14
    - defining 6-39
    - display indicator 6-37
    - filing code 18-55
    - global storage 16-13
    - global structure 16-14, 16-19, 20-4
    - %linenum variable 10-9
    - map data specifications 16-19, 18-47
    - maximum length 6-37
    - maximum line count 6-41
    - One Per Node global storage 6-38, 16-14
    - referencing 6-39
    - relational model 2-23
    - storage modes 6-37
  - Multiple Choice data type 6-16
    - conversion code 10-4, 10-6
    - conversion/validation code 10-3
    - defaults 6-9
    - existing data 6-16
    - internal values 6-16
  - Multi-row form 3-5
    - GETOUT key 3-11
  - \$mux command 3-2
  - ^mxdd global 20-2
    - access path specification node 20-19
    - base table node structure 20-5
    - basic structure 20-2
    - creating 20-2
    - first subscript 20-3
    - map nodes 20-17
    - multi-line field node 20-4
    - nodes 20-3, 20-6
    - privileges 20-30
    - required and default values 20-3
    - row ID specification nodes 20-22
    - structure 20-2
    - subscript values 20-3
    - views 20-25
  - ^mxdd import file
    - See ^mxdd global
- ## N
- N variable 6-26
  - Name data type 6-18
    - collating sequence 6-28, 6-31
    - conversion code 10-4, 10-6
    - conversion/validation code 10-3
    - defaults 6-9
    - index maps 16-24
    - validation code 10-7
  - Name subscript 20-3

National language Glossary-7  
help and error message translations 13-4

%new variable 10-9

%newext variable 10-9

\$\$NEXT function 16-4  
assigns Row ID values 16-5  
Row ID node 20-11

NEXT subroutine 18-31  
access level values 18-34  
defining override 18-35  
reactivating default 18-35  
where to define 18-16

Node field values 18-46  
required 20-4

Node subscript 20-3

Notes  
See Comments

Number data type 6-20  
conversion code 10-4, 10-6  
conversion/validation code 10-3  
defaults 6-9  
validation code 10-7

NUMROWS reserved word 6-35

## O

Object code Glossary-7

ObjectType subscript 20-3

%ok variable 10-9  
base table validation 10-16  
in conversion code 10-6

%old variable 10-9

%oldext variable 10-9

One Per Node storage mode 6-38, 6-41, 16-13

One-way outer joins 2-16, 7-3

On-line help 3-11  
for fields 13-1

Open M Developer 1-2, 1-4, 3-3  
See also Data Dictionary, FileMan  
interface, Form Generator,  
Query Editor  
access privileges 12-2

privileges menu 5-21, 14-23, 14-24  
privileges option 5-23

Open M Developer Menu 3-3

Open M with SQL 1-2  
accessing the relational database 2-9  
defining a database 1-7  
extensions to relational model 2-16  
open systems architecture 1-5  
relational database 2-1  
schemas 2-8

OPENM.DAT 2-8, Glossary-7

ORDER BY clause  
collation sequence 6-31  
example 8-24  
override lookup query 8-19  
query options 8-20  
transformation functions affect 8-22

\$ORDER function 18-30  
NEXT subroutine 18-31, 18-34

Outer join  
designative reference 7-3

Output device 15-3

Output only protection 6-47

Override filing code 16-17, 17-19  
defining 18-55  
defining override 18-56  
for all fields 18-55  
for selected fields 18-55

Override help 13-5  
defining 13-8  
replaces Long Help message 13-3  
window 13-8

Owner  
of base table 5-10, 5-23, 12-2  
of view 14-21, 14-23, 14-24

## P

^\*parent global 5-13, 5-15

Parent reference field 16-6, Glossary-7  
column number 6-47  
in master map 16-13

Parent table 2-19, 5-11

- 
- arrow syntax 2-21
  - Piece
    - access type 18-16, 18-21, 18-22
    - data access expression 18-19
    - delimited 17-8, 18-27, 18-45
    - listing report 15-15
    - undelimited 17-8
  - Piece counter field 17-12, 18-23, 18-27, 18-30
    - incrementing 18-30
  - \$PIECE function 18-19, 18-21
  - PREVIOUS key 3-10
    - unsaved data options 3-14
  - Primary key Glossary-7
    - See Row ID field
  - Printing
    - specifying device 15-3
  - Privileges Glossary-8
    - defining in ^mxdd import global 20-30
    - granting 12-2
    - to copy a view 14-28
    - to edit a view 14-23
  - Privileges menu 5-21, 5-23, 14-23, 14-24, 20-30
  - PROCEED key 3-10
    - save toggle 3-15
    - unsaved data options 3-14
  - Programmer mode Glossary-8
  - Project operation 2-12
  - Prompt Glossary-8
- Q**
- Query 2-9, Glossary-8
    - compiling 17-18
    - computed field 6-42
    - copy another view's query 14-19
    - copy into view 14-18
    - index map for 9-4
    - override lookup query example 8-23
    - query-based view 14-17
    - query-based view and DDL 14-20
    - SQL SELECT statement 2-9
    - using to test customized mapping 17-14
    - view 14-7
  - Query Editor 14-18, 14-20
  - Query Generator 14-18
  - Query Optimizer 1-15
  - Query Options window 8-20
- R**
- Record
    - See Row
  - Referenced table 6-15
  - REFERENCES privilege
    - for base table 12-2
  - Referential integrity 1-16, Glossary-8
  - Relation Glossary-8
  - Relational database Glossary-8
    - accessing data from 2-9
    - as view of existing M database 1-8
    - characteristics 2-2
    - define in Data Dictionary 1-7
    - design 17-5
    - designing 4-2
    - fields 2-2
    - first normal form 2-6
    - for non-ISM systems 2-8
    - mapping links to M globals 1-13
    - overview 1-2, 2-1
    - querying via SQL 2-9
    - referential integrity 7-8
    - relation implemented as table 2-2
    - structure of 2-2
    - tables 2-3
    - viewing M database as 17-1
  - Relational model 2-1
    - first normal form 2-6
    - implicit joins 2-18
    - InterSystems extensions 2-16
    - one-way outer joins 2-16
    - schemas 2-8
    - support for multi-line fields 2-23
  - Relational operators 2-10
    - cartesian product 2-10
    - joins 2-14
    - project 2-12
    - restrict 2-13

- Relational server 1-5
  - Reports Glossary-8
    - See M/PACT
    - data dictionary definitions 15-1
    - fields listed by column number 15-6
    - fields listed by name 15-5
    - list of base tables report 15-4
    - list of views 15-9
    - listing fields in a base table 15-5
    - listing fields in a view 15-11
    - listing mapping 15-15
    - printing 15-3
    - series of base tables 15-7
  - Required field Glossary-9
    - defining 6-47, 6-49
    - maybe option 6-47
    - only upon condition 6-49
  - REQUIRED node 20-4
  - Restrict operation 2-13
  - RETURN key 3-10
  - Right Arrow key 3-10
  - Routine name prefix 12-4
    - defining 5-17
  - Routines
    - base table compile 5-17
    - deleting 5-18
    - routine name 5-17
    - specifying 5-10
  - Row 2-2, Glossary-9
    - approximate number in base table 5-8
    - delete messages 12-7, 12-9
    - insert messages 12-7, 12-8
    - locking/unlocking 5-10
    - message display logic 12-11
    - message translations 12-9
  - Row ID field 2-3, 17-13, Glossary-9
    - as data field 17-32
    - based on childsub field 16-6, 16-8
    - based on delimited piece 17-32
    - based on parent reference field 16-6, 16-8
    - based on piece counter field 17-12
    - child table example 16-6
    - column number 6-47
    - conversion code 10-5, 10-6
    - conversion/validation code 10-3
    - customized physical structure 17-10, 18-9
    - default physical structure 5-16
    - default value 6-51
    - defaults 6-9
    - definition 16-2
    - examining default definition 16-3
    - fields used to specify 16-16, 18-9
    - foreign key in designative reference 7-2
    - global storage 16-12
    - identifying in ^ACCT global example 17-26
    - identifying in ^FLAVORS global example 17-31
    - identifying in ^ZODIAC global example 17-20
    - identifying in existing M database 17-7
    - in child tables 16-5
    - in master map 16-13
    - index maps 9-3
    - piece counter 17-12
    - row ID specifications 16-20
  - Row ID Specifications 18-10
    - default physical structure 16-20
    - defining in ^mxdd global 20-22
    - deleting 18-10
    - editing 18-43
    - invalidated 18-25
  - Row selection mode 3-8
  - %row variable 16-17, 16-19, 18-9
    - default filing code 18-55
    - full row reference 18-20, 18-41
    - node field entry 18-45
    - node field value 18-46
    - SET command 18-52
  - Runtime environment
    - customize via on-line help 3-11
- ## S
- ^SAMPLE global example 17-2, 18-2
    - data 17-5
    - enhanced 17-19
    - fields 17-7
  - Save and Remain key 3-10



- Save Data menu
  - on exit from form 3-14
  - PROCEED options 3-14
- Save Menu 3-14
- Save on PROCEED menu
  - toggle 3-15
- Schema 2-8
- ScreenMan application generator 19-7
- Security
  - See Privileges
- SELECT privilege
  - copying a view 14-28
  - for base table 12-2
- SELECT statement 2-9
  - example 2-21
  - field display order 6-47
  - index map example 18-24
  - index maps 18-24
  - loop access 18-30
  - multi-line fields 2-23
  - query options 8-20
  - single value access 18-29
  - views 14-4
  - with collating function 18-23
- Selecting mode 3-8
- Server
  - See Relational Server
- SET command
  - additional validation code example 10-8
  - external-to-internal conversion example 10-5, 10-14
  - index map structure 18-5
  - internal-to-external conversion example 10-6, 10-14
  - master map structure 16-14, 18-5
  - multi-line fields 16-14
  - set global value 1-3
- Single-row form 3-5
  - GETOUT key 3-11
- Sort
  - See Collation sequence
- Sorting data
  - index maps for 9-4
- Special Access Code 18-16
  - additional data variables 18-31
  - data access variables 18-39
  - default invalid values 18-32
  - defining first and last valid values 18-32
  - specifying 18-29
  - types 18-31
- SQL 1-3, 2-9, Glossary-9
  - affected by field definition 6-2
  - arrow syntax 2-21, 7-7
  - automatic code generation 1-15
  - colon syntax 8-23
  - comparison operators 8-22
  - computed fields 6-44
  - embedded 6-44, Glossary-3
  - InterSystems extensions to 2-16, 2-21, 8-21
  - lookup query example 8-24
  - multi-line fields 2-23
  - query-based view DDL statements 14-20
  - querying the relational database 2-9
  - relational operations 2-10
  - reserved words 6-5, C-1
  - %STARTSWITH operator 8-21
  - user input variables 8-23
- ##sql 6-44
- &sql 6-44, Glossary-3
- SQL statements
  - AND clause 2-21
  - DISTINCT clause 8-20
  - FROM clause 2-21, 2-22, 8-20, 14-3
  - GROUP BY clause 8-20
  - HAVING clause 8-20
  - INSERT statement 6-41
  - ORDER BY clause 8-20
  - SELECT statement 6-47, 8-20, 18-23
  - UPDATE statement 6-41
  - WHERE clause 2-21, 2-22, 8-20
- Starting Table of view 14-7, 14-8
  - changing 14-23, 14-25
  - in another view 14-3
- %STARTSWITH operator 8-21
- STORE error 6-2
- sub
  - See Access Type

### Subscript

- abbreviations 20-3
- access level 18-9
- access type 18-16
- constant value 17-8
- defining 18-14
- in master map 16-12
- in ^mxdd import global 20-2

### Sub-view

- join specifications 14-11

### System Help Menu 3-11

### System Manager

- assigns privileges 5-21, 5-23, 14-23, 14-24
- sets date format default 6-12
- sets full year display default 6-13
- sets global name default 5-13
- sets save menu toggle 3-15

## T

### T variable 6-12

### TAB key 3-10

### Table

- See Base table, Child table, Parent table, View, Virtual table

### ^temp 12-5

### Terminal types B-1

- Keyboard mapping B-1

### Text data type 6-24

- collating sequence 6-28, 6-31
- conversion code 10-5, 10-6
- conversion/validation code 10-3
- defaults 6-9

### Time data type 6-25

- conversion code 10-5, 10-6
- conversion/validation code 10-3
- data type errors 6-10
- defaults 6-9
- importing data 6-54
- midnight 6-26
- n variable 6-26
- validation code 10-8

### \$TRANSLATE function 10-14

### Translations

### defining error messages in ^mxdd

- import file 20-13
- defining help messages in ^mxdd import file 20-14
- field help and error messages 13-4
- languages supported 12-10, 13-4

### Triggers 1-8, 1-9, 1-16, 4-3, 11-1, Glossary-9

- action type 11-12
- characteristic relationship 5-21, 7-5, 11-14
- conditional execution 11-7
- %data variable 11-9
- defining filing code for a field 11-4
- defining in ^mxdd global 20-15
- defining override filing code 18-55
- %edit variable 11-9
- entering filing code 18-46
- examples 11-15
- execution conditions 11-7
- form triggers 11-2
- from FileMan indices 19-7
- M code 11-6
- multiple trigger items 11-8
- override filing code 17-19
- overview 11-2
- post-filing 11-5
- pre-filing 11-4
- SQL code 11-5
- subscript values 20-15
- trigger item 11-7
- unconditional execution 11-7
- variables in code 11-8

### Tuple

- See Row

## U

### Unique field 6-7, 6-34

- index map for 6-34, 9-4, 16-22

### UPDATE filing code field 18-56

### UPDATE privilege

- for base table 12-2

### UPDATE statement

- field default value 6-51
- multi-line field line count 6-41
- pre-filing DELETE trigger 11-19

UPDATE trigger  
   field value 11-8  
   pre-filing 11-4, 11-5, 11-9, 11-17  
   pre-filing example 11-18

UPPER 6-30, 8-16, 8-22

\$\$UPPER function 18-24  
   index maps 18-23

%UPPER SQL extension function 8-22

%urdel utility 5-18, 5-19

## V

%val variable 10-6, 10-9  
   additional validation code 10-8  
   data conversion 10-5  
   in field conversion code 10-13

Validation code 6-7, Glossary-10  
   accessing a row 18-29  
   additional validation code 10-7  
   automatically generated 10-3  
   base table level 1-16, 10-15  
   computed fields 6-43  
   \$DATA function 18-31  
   defining first and last values 18-32  
   field length 12-5  
   field level 1-16, 10-7  
   first/last subscript values 18-16  
   on external value 10-8  
   overview 10-2  
   variables allowed 10-9

Value Expression 18-16  
   access level 18-22  
   functions and index maps 18-23

V FName subscript 20-3

View 1-12, 14-1, Glossary-10  
   advanced options 14-21  
   argument to FROM clause 14-3  
   benefits 2-5  
   changing starting table 14-25  
   characteristic relationship 14-8  
   comments 14-22  
   convert query-based to table-based 14-20  
   copy another view's query 14-19  
   copy existing query into 14-18

  copying 14-28  
   creating 14-5  
   defining in ^mxdd global 20-25  
   deleting definition 14-24  
   description 14-2  
   designative reference 14-8  
   duplicate field names 6-5  
   editing definition 14-23  
   field name 14-12  
   fields 14-14, 14-15  
   fields to include 14-11, 14-27  
   implicit joins 14-14  
   implicit joins link tables 14-9  
   join specifications 14-7, 14-8  
   listing all 15-9  
   listing fields in a view 15-11  
   M/PACT report source 14-3  
   of FileMan file 19-23  
   overview 2-5  
   owner of 14-21  
   query-based 14-7, 14-17, 14-20  
   report on definition 15-9  
   report sorted by fields 15-11  
   report sorted by join path 15-9  
   source for M/PACT report 14-2  
   starting table 14-3, 14-23  
   sub-view 14-11  
   table-based 14-7, 14-8  
   types 14-4  
   uses of 14-3

View Definition master window 14-6

View Join Specification window 14-13

Virtual field Glossary-10

Virtual table 1-7, 1-12, 2-3, 2-4, 14-2,  
   Glossary-10  
   See also View, Query

V FName subscript 20-3

## W

WHERE clause 2-21  
   example 2-21, 8-24  
   override lookup query 8-19  
   query options 8-20

Window 3-7, Glossary-10  
   auxiliary Glossary-1

- caption Glossary-4
- data entry 3-8
- horizontal options menu 3-8
- master 3-7, Glossary-6
- row selection Glossary-9

## Y

- Yes/No data type 6-27
  - conversion code 10-5, 10-6
  - conversion/validation code 10-3
  - defaults 6-10
  - null value 6-27
  - validation code 10-8

## Z

- \$ZDATE function 10-6
  - computed field calculations 6-42
- ^ZODIAC global example 17-19