
Open M with SQL — Version F.11

Release Information

Version: Open M with SQL F.11

Date: October 22, 1997

Part Number

IS-SQL-0-F.11A-CP-R

Open M with SQL — F.11 Release Information

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™, InterSystems Cache™, 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 Corporation.

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

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

Developers: +1 (617) 494-0867

Table of Contents

Release Information

Converting Your Open M with SQL Applications to Version F.11	5
New Features and Enhancements	8
Corrections	15
Limitations	19
Documentation Notes	24

A Converting Your Applications to the New Global Structure for Routines

New Global Structure for Routines	A-1
ROUTINE Global Conversion Utility	A-3
Setting the Number of Backup Versions	A-4

B InterSystems SQL DDL Grammar

Conventions used in DDL Grammar	B-1
Create Table Command	B-2
Alter Table Command	B-9
Drop Table Command	B-11
Create Index Command	B-12
Drop Index Command	B-13
DDL Error Messages	B-14

Release Information

Converting Your Open M with SQL Applications to Version F.11

When you upgrade to Open M with SQL Version F.11, you need to perform the following steps to convert your existing Open M with SQL applications:

1. Back up your system. Once you have converted to the new routine storage global structure, you cannot downgrade to earlier versions of Open M with SQL.
2. Run the ^ROUTINE global conversion utility, as described in Addendum A, *Converting Your Applications to the New Global Structure for Routines*.

Warning: You must convert applications to the new global structure for routines prior to running Open M with SQL Version F.11. Serious database problems may occur if this conversion is not run.

3. Run the Open M with SQL conversion program in each directory/UCI (except the manager's directory and the Open M with SQL common directory). The conversion program consists of a series of routines that update certain Open M with SQL internal structures to make them compatible with the new version.

Open M with SQL provides the Conversion Manager utility to assist you in running the appropriate conversion program(s).

4. After running the conversion program, you should recompile all Open M with SQL objects in each directory/UCI (except the manager's directory and the Open M with SQL common directory).

Open M with SQL provides the %mcompil utility to assist you in recompiling Open M with SQL objects.

Conversion Manager Utility

The Conversion Manager utility runs conversion routines beneath a window-based interface. It automates the task of converting to a new Open M with SQL version on a per directory basis and allows you to selectively exclude undesired features of the target version. The Conversion Manager also allows you to run conversion routines multiple times and print reports associated with a particular conversion task.

You should normally run the Conversion Manager in each directory/UCI that you are upgrading.

Note: You must run all required conversion tasks for each upgrade level. For example, if you are upgrading from Open M with SQL Version F.7 to Version F.11 you must run the conversion manager for F.* to F.10 and F.10 to F.11.

For complete information on how to convert Open M with SQL applications using the Conversion Manager utility and recompile Open M with SQL objects using the %mcompil utility, see the *Open M with SQL Database Administrator's Guide*.

F.* to F.10 Conversion Tasks

If you are upgrading from an Open M with SQL F release prior to F.10, run the F.* to F.10 conversion tasks from the Conversion Manager Menu.

```
ÚAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAConversion
ManagerAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA;
3 NOTE: Some conversions may not be necessary if they do not apply to a      3
3 DBMS software upgrade.                                                    3
3                                                                            3
3 From Version      To Version                                             3
3 B                C                < Conversion Requirements > 3
3 C                D                < Conversion Requirements > 3
3 D                E                < Conversion Requirements > 3
3 E                F                < Conversion Requirements > 3
3 F.*              F.10            < Conversion Requirements > 3
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAU
Conversion Manager                                                    Press <Help> For Help
```

The F.* to F.10 conversion includes the following new conversion tasks:

Table 1: F.*-to-F.10 Conversion Tasks

Conversion Task	Description	Recommendation
Reserve Word/Identifier Conflict	Reports tables, views and fields with names that are SQL reserved words. These objects must be renamed to avoid conflicts.	This conversion task is required—you must run it.
Add Validation Code to Multiple-Choice Fields	Adds validation code for each multiple-choice field defined. If a multiple-choice field is set to an invalid value via an INSERT or UPDATE (or from a form), field validation fails and an error is reported.	This conversion task is required—you must run it.
Update Base Table Metadata	Updates Base Tables to accommodate enhanced default structure.	This conversion task is required—you must run it.

F.10 to F.11 Conversion Tasks

InterSystems has added an option for F.10 to F.11 conversion tasks to the Conversion Manager Menu. This option includes the following conversion task:

Table 2: F.10-to-F.11 Conversion Tasks

Conversion Task	Description	Recommendation
Validate & Repopulate General View Fields	Checks all query-based views and reports any incorrectly defined views. Populates new field information for non-atomic fields (e.g. expressions, aggregates, functions) in query-based views.	Not required. Highly recommended if you use general views.

New Features and Enhancements

InterSystems SQL Data Definition Language (DDL)

InterSystems now supports Data Definition Language (DDL) for the following statements:

- n CREATE TABLE
- n ALTER TABLE
- n DROP TABLE
- n CREATE INDEX
- n DROP INDEX

Complete DDL grammar for the above statements is documented in Addendum B, *InterSystems SQL DDL Grammar*.

InterSystems SQL DDL allows you to export databases from other applications into Open M with SQL. DDL also allows you to create, modify and delete tables, and create and delete indexes via embedded SQL in your M code routines.

DDL statements are compiled into M code and executed when the M code is run. The execution only affects the directory in which the statement is run and refers to the data dictionary at run time.

The following criteria must be satisfied to execute DDL statements:

- n The Open M with SQL user must be logged into Open M with SQL with a valid username and password — \$p(%msql,\$c(1)) must be equal to a valid username.
- n The Open M with SQL user must have Data Dictionary Access turned on in its user definition.

DDL Default Base Table Options

The SQL / Base Table Options Window on the System Configuration Settings Screen allows you to set system defaults for DDL DROP TABLE and DROP COLUMN.

Table 3: SQL / Base Table Options for DDL Statements

Option	Meaning
Does DDL DROP TABLE Statement Delete Data?	Enter 'Yes' to delete the table's data when a table is dropped.
Does DDL DROP COLUMN Statement Delete Data?	Enter 'Yes' to DELETE the column's data when a column is dropped

- ## System Configuration

Relational Client/Server

- n Open M with SQL Version F.11 includes the InterSystems Open M Relational Server. The Open M Relational Server supports the following InterSystems Relational Client releases:
 - Open M Relational Client Version 1.95
 - Open M Relational Client Version 2.x
- n The Open M Relational Server supports the ODBC outer join syntax:

ODBC Syntax	Shorthand Syntax
--(*vendor(Microsoft),product(ODBC) oj <outer-join>*)	[oj <outer-join>]
<i>where:</i> <outer-join> ::= <table reference> LEFT OUTER JOIN { <table reference> <outer-join> } ON <search condition>	

- n The Relational Server supports the ODBC escape syntax for DATE, TIME, and TIMESTAMP fields as follows:

ODBC Syntax	Shorthand Syntax
DATE: --(*vendor(Microsoft),product(ODBC) d 'YYYY-MM-DD' *)--	{d 'YYYY-MM-DD'}
TIME: --(*vendor(Microsoft),product(ODBC) t 'HH:MM:SS' *)--	{t 'HH:MM:SS'}
TIMESTAMP: --(*vendor(Microsoft),product(ODBC) ts 'YYYY-MM-DD HH:MM:SS' *)--	{ts 'YYYY-MM-DD HH:MM:SS' }

- n Improvements to stored procedure information are included in the Relational Server.
- n The Relational Server error log can now be exited by typing “^”, “q” or “Q” at the “Delete this error from the log?” prompt.
- n The SQL parser on the server removes leading underscores from identifier names (e.g. tables, fields) and removes identifier qualifications. These tasks were formerly performed by the relational client.

The SQL parser now accepts qualifications in the form:
 <owner>.<catalog>.<schema>.<tablename>.<fieldname>

- n SQL_TIMESTAMP and MEMO (SQL_LONGVARCHAR) are supported as data types for the Relational Server. These internal definitions allow fields of these types to be compared, sorted, etc., without error.

Procedure Defining an SQL_TIMESTAMP field:

1. Create a field in the Data Dictionary via the (Field Definition) form.
2. Specify Number as the field's Data Type.
3. Select Advanced Options from the horizontal menu and specify the value SQL_TIMESTAMP for the field This field is a Date data type in. This creates appropriate code for:
 - Internal to External Code
 - External to Internal Code
 - Validation Code
 - Error Message

The external format for the SQL_TIMESTAMP data type is:
YYYY-MM-DD HH:MM:SS

The internal value is an (INTEGER) defined in the following way:
<date-internal-\$h-format>*86400+<time-internal-\$h-format>

The data type number that the server sends to the client for SQL_TIMESTAMP fields is 20.

Procedure Defining a MEMO (SQL_LONGVARCHAR) field:

1. Create a field in the Data Dictionary via the Field Definition form.
2. Specify Text as the field's Data Type.
3. Specify a Maximum Length that represents the maximum length of an instance of a Multi-Line field.
4. Specify Yes as the value for the Multi-Line Field field.
5. In the Multi-Line Field Options window, specify a number to represent the value for the Maximum number of lines to allow field.

By specifying maximum length and maximum lines, the total maximum length of the field sent to the client is calculated in the following way:

<maximum length>*<maximum lines>+1

If no value is specified for maximum lines, then the total maximum length of the field sent to the client defaults to (32K+1) or (32768).

The data type number that the server sends to the client for MEMO fields is (108).

- n The LOGIC data type (SQL_BIT) is supported for Visual Foxpro version 5 clients communicating with the Relational Server. These internal definitions allow fields of these types to be compared, sorted, etc. without error.

Procedure Defining a LOGIC field:

1. Create a field in the Data Dictionary via the Field Definition form.
2. Specify Number as the field's Data Type.
3. Select Advanced Options from the horizontal menu and specify LOGIC for the field.

This creates appropriate code for:

- Internal to External Code
- External to Internal Code
- Validation Code
- Error Message

The internal & external value for the LOGIC data type is always a single byte integer with one of the following values:

Value	Meaning
0	False
1	True

The M/SQL data type number that the server sends to the client for fields of type LOGIC is 21.

Implementation of SQL

- n The %NOLOCK operator allows INSERT/UPDATE/DELETE functions to execute without the overhead of locking the table, i.e. during benchmark testing. The syntax is as follows:

```
INSERT [%NOCHECK] [%NOLOCK] ...  
UPDATE [%NOCHECK] [%NOLOCK] ...  
DELETE [%NOCHECK] [%NOLOCK] ...
```

Note: All tables that use this new feature must be recompiled.

Example

```
INSERT %NOCHECK %NOLOCK INTO Accounts (Notes) VALUES ('Current')  
UPDATE %NOCHECK %NOLOCK Accounts SET Notes = 'New Customer' WHERE Order = 1  
DELETE %NOCHECK %NOLOCK FROM Accounts WHERE Order = 0
```

- n The Oracle function SIGN is supported as follows:

SIGN(variable)

If the variable is:	SIGN(variable) returns:
positive	1
negative	-1
zero	0
non numeric value (text) in a numeric field	0
alpha value (date, time, text) in a text field	""

Example The following statement selects all customers with a positive balance:

```
SELECT Customer FROM Accounts WHERE SIGN(Balance) = 1
```

Data Dictionary

- n The “List of Base Tables” option from the Reports on Data Dictionary menu displays information on the compilation status of each base table. This field shows either the last compilation was Successful, Failure, or Uncompiled.
- n The \$\$next() function is enhanced to use the \$ZINCR function to increment global counters on platforms that support \$ZINCR. The \$ZINCR command increments the counters faster and does not need to lock the global.

Note: Since \$ZINCR can only be executed on global variables, \$\$next() may no longer take a local variable or local array node as an argument.

User Utilities

- n The Change Password Utility allows the Open M with SQL system manager to set up an account for a new user. The user can now change his/her password confidentially.
- The Open M with SQL system manager (SYSTEM) can change any user’s password without knowledge of the previous password.
 - The new password must be entered twice to verify the change.
 - The user password is now displayed on the User Security Definition form in block-out mode for enhanced security.

General Import/Export

- n Performance improvements in searching for Base Table related objects are included in the General Export/Import utility.

Corrections

Data Dictionary

- n If an error is encountered using the `$$comptab^%msqlapi` function the table is unlocked.
- n InterSystems has corrected a problem that caused computed fields that were not always equal to the computation to be recomputed at filing time.
- n InterSystems has corrected a problem that caused a loop if a map had more than nine subscript levels.
- n InterSystems has corrected a problem copying views based on queries if the queries used the AS operator in the SELECT list.
- n Row ID fields for default structure tables (including those created via DDL) are automatically hidden from ODBC clients. You can specify fields as hidden by setting the “Exclude from lookup lists?” field to “Yes”.
 - Any fields designated hidden will not appear on a query of the form `SELECT * FROM TABLE`, but you can explicitly select hidden columns by name.
 - You can not perform an INSERT or UPDATE on hidden fields unless they are specified in the column list. If no column list is specified, you may not provide values in the VALUES clause for hidden fields, and any mismatch between the number of values in the VALUES clause and the number of non-hidden columns generates an SQL error.
- n InterSystems has corrected an UNDEFINED error defining view-based views.
- n If a view is based on a query that has aggregate functions, the message “Aggregate functions are not allowed here” is displayed after the query is entered and the PROCEED or SAVE key is pressed.
- n InterSystems has corrected an UNDEFINED error with complex conditional maps.
- n InterSystems has corrected the default error message text for date fields.
- n InterSystems has corrected a problem in the View Definition form where a field would not appear in the view fields list if the field was already in another join specification on the same view.
- n InterSystems has corrected a SYNTAX error compiling a child table.

Forms

- n InterSystems has corrected a problem where the improper value of a designated display field was used when comparing the contents of the field on triggers if the designated reference field did not exist on any of the form's windows.
- n InterSystems has corrected an UNDEFINED error using the "Maximum number of rows to fetch per cycle" option.
- n The {%filetype} variable returns "INQUIRY" for forms in inquiry mode.
- n InterSystems has corrected an UNDEFINED error in table validation code during form lookups.
- n InterSystems has corrected a SYNTAX error returning from a pop-up menu to a menu bar attached to a form.
- n InterSystems has corrected an UNDEFINED error returning to a form from an attached menu that calls a query.
- n InterSystems has corrected an UNDEFINED error compiling a multi-row form containing several windows with "Allow Insertion of New Rows" set to "No".
- n InterSystems has corrected a problem with the {%return_action} variable. It now accurately reflects the last keystroke hit in a multi-row form.
- n The GETOUTALL option in forms exits out of all forms when selected at any point in the application.
- n InterSystems has corrected an UNDEFINED error compiling a form with validation code, field length validation enabled, and no English error messages defined.

Queries

- n A query with a view in the FROM clause now uses the field description from the view in the query output.
- n Aliases specified in the AS clause of interactive, defined, and base queries for views are used as column titles instead of the original field description.

Menu Objects

- n InterSystems has corrected a problem where a menu bar invoked from a pre-window trigger on non-displayed window appeared at the top of the screen rather than at the bottom.
- n InterSystems has corrected a SYNTAX error exiting a menu bar attached to the master window of a window ordered form.

- n InterSystems has corrected a problem where calling a variable placement window from a menu bar without passing a caller id could result in a PARAMETER error at run time.

Import/Export

- n If the display length for a lookup field is less than the header, import trims the header to fit and no error occurs.
- n InterSystems has corrected a bug in import where a form field display that was out of range would cause an SQLCODE=104 error.
- n Export now exports more than one data selection in a report.
- n InterSystems has corrected a problem with exporting view-based reports with detailed calculated fields.
- n InterSystems has corrected a problem recognizing table aliases in the from clause of queries during import.
- n InterSystems has corrected a problem where the import of a table with its Row ID based on other fields and a default external value would lose the default external expression.
- n InterSystems has corrected a problem where import deleted views based on copied queries when the tables referenced in the FROM clause were re-imported, rather than revalidating these views.
- n InterSystems has corrected a problem importing forms with fields that could not be mapped. These fields are now removed cleanly and the rest of the form fields are imported properly.

Relational Client/Server

- n InterSystems has corrected an UNDEFINED error when using the Oracle DECODE function without specifying a default value.
- n InterSystems has corrected an error, 25188 — Unable to write to server, trying to attach a table from Microsoft Access against a DTM server in a DTM network.
- n The Stored Procedure matching algorithm works properly with SQL statements containing literals.

Privileges

- n The lookup windows in the Grant and Revoke Privilege forms now display role names.

Macro Routine Programming

- InterSystems has corrected a problem where routines with 30 or more embedded SQL statements could generate duplicate line tags.
- The “Break 1” instruction was removed from the %urprint routine for non-ISM platforms.
- Macro definition expansion has improved compile time syntax checking.

Full Screen Editor

- InterSystems has corrected a problem where a routine selected in the “Insert Routine” option in the Buffer menu of the full screen routine editor, would remain locked after exiting the editor.

Reports

- InterSystems has corrected an UNDEFINED error calling a report via report^%msql(report,,device,format) from an old style form.

Terminal Types

- InterSystems has corrected an UNDEFINED error defining custom terminals in MSM using arrow, PF, and F keys.

Utilities

- The Integrity Check Utility checks for zero values in maximum field length for base tables, and in maximum field length and display length for forms.
- InterSystems has corrected a problem in the FileMan - Open M Developer Linker utility. If a file and the file indexes were updated and the file contained a MULTIPLE index which did not have standard SET and KILL code, an error would occur. The index was not properly translated to post-filing INSERT and DELETE triggers in the base table definition.

Limitations

Open M with SQL (General)

- Open M with SQL objects (with the exception of forms) can not have foreign characters in their identifier names. Form names, however, can include both foreign characters and punctuation marks in their identifier names. [M/SQL C.91-01]

Data Dictionary

- When running Open M with SQL on an ISM or DSM system, a Base Table can contain a maximum of approximately 150 fields.

When running Open M with SQL on a DTM system, a Base Table can contain a maximum of approximately 100 fields. This limitation is due to the fact that DTM imposes a 32KB limit on routine size. InterSystems recommends that you keep the number of fields per Base Table as small as possible to assure best results. [Open M/SQL Version F.8]
- The calculation for a computed field can reference a maximum of approximately 35 other fields. [M/SQL 89-12]
- A single lookup query defined in the Data Dictionary lookup specifications should contain no more than 7 lookup fields and 7 lookup display fields. From this, you should subtract one field for each level of dependency of the table, i.e. 6 fields for a child table, 5 for a grandchild table, etc. [M/SQL 89-12]
- InterSystems discourages use of the %data and %edit arrays in SQL triggers due to the possibility of their being NEWed within the SQL trigger. If necessary, use a previous trigger item to copy the values from %data and %edit into local variables. [M/SQL 89-12]
- If you change a RowID definition, Open M with SQL does not automatically update the Row ID calculation in the Data Dictionary maps. In order to force the recalculation of the RowID, you must delete the existing RowID calculation from every map definition in the table. To do this, enter each map definition, select RowID Calculation, and delete the existing Row ID calculation by pressing the <REMOVE> key or by selecting the Delete an Entire Row option from the horizontal menu. Once you have deleted the existing calculation, Open M with SQL automatically generates the new calculation. [M/SQL 89-12]

Form Generator

- n Multi-row forms cannot be window-ordered. If you attempt to create a multi-row window-ordered form, the Form Compiler issues a warning message during compilation, and the form ignores the window-order list at run time. [Open M/SQL Version F.6]
- n Database fields with the data types Designative Reference and Multiple Choice do not display horizontal menu options. [Open M/SQL Version F.6]
- n If you have a RowID that is based on other fields and those based-on fields appear on a form, you must make sure that users do not modify their values after filing an initial value. Modifying the values of based-on fields may cause the form to behave erratically. [Open M/SQL Version F.6]
- n Field validation code does not act on default values for form-only fields or on default values passed in via a default array (the 8th parameter of the M form call syntax). You are responsible for your own validation checking on these default values. [Open M/SQL Version E.3]

Branching Fields

- n It is not possible to specify Conversion/Validation code for fields of data type Branching. The Conversion/Validation Code option does not appear on the horizontal options menu of the field definition. [Open M/SQL Version F.6]

Triggers

- n Open M with SQL enforces the following limitations for Post-Window triggers associated with the master window of a multi-row form:
 - The action type Set Field is not available
 - You cannot reference fields using curly brace syntax {fieldname} in triggers of action type M Code [Open M/SQL Version F.7]
- n The Delete Row action type is not supported at the following trigger locations for multi-row forms:
 - Pre-Window triggers
 - Post-Window triggers
 - Post-Retrieval triggers [Open M/SQL Version F.7]

- n When using Set Field triggers to target Designative Display fields, you cannot set the field directly. On the Set Field popup window, you must accept the *No* response to the “Set the Field Directly?” prompt. Attempting to set the field directly causes the Form Compiler to generate a warning message during form compilation stating that the trigger is ignored. [Open M/SQL Version F.6]

Programmed Lookups

- n It is not possible to define programmed lookups for fields with data types Branching, Designative Display, Designative Reference, or Multiple Choice. Selecting the Programmed Lookups option on the horizontal options menu of the field definition returns you to the main field definition window. [Open M/SQL Version F.6]
- n You can not reference the variables *x*, *y* or *tmp* in programmed lookup code. If you reference any of these variables, your programmed lookup does not work properly. [Open M/SQL Version F.6]

Roll and Scroll Mode

- n You cannot access the M/SQL System Help Menu when running forms in Roll-and-Scroll mode. Pressing the <HELP> key while in Roll-and-Scroll mode causes the terminal to beep. [Open M/SQL Version F.6]
- n You cannot see or access any menu objects when running forms in Roll-and-Scroll mode. This includes form menu bars and window menu bars as well as menu bars and pop-up menus called by form triggers. [Open M/SQL Version F.6]

Lookup Specifications

- n In the Lookup Specification window (both in the Data Dictionary and in the Form Generator), you must not specify multi-line fields as a lookup fields (Fields To Lookup On) or lookup display fields (Fields To Display). [Open M/SQL Version F.6]
- n Lookup boxes can accommodate a maximum of 7 lookup display fields. If you define more than 7 lookup display fields in a lookup query, Open M with SQL displays only the first 7. You receive a warning message to this effect at compile time. [Open M/SQL Version E.5]

Menu Generator

- n The Menu Generator limits Menu Item names for old-style vertical and horizontal menus to 42 characters. [M/SQL B 90-09]

M/PACT

- n M/PACT reports cannot use {fieldname} reference in pre- and post-report triggers with action type of M Code. [Open M with SQL Version F.11]

- n When an M/PACT report is based on either of the following:

- Query
- View that is itself based on a query

it cannot use the “Eliminate Cartesian Product?” option.

In this case, the “Eliminate Cartesian Product?” field on the Report Definition Advanced Options window is set to *No*, and you cannot access it. [Open M/SQL Version F.6]

- n In a report based on a query data source, once the query is imported into the report definition, it is not possible to modify the text of the original query and reimport it or otherwise update the query within the report. [Open M/SQL Version E.5]
- n M/PACT places a limitation on the number of data columns that you can include in a report definition. If you exceed this limitation, compilation of the report definition fails due to a <MAXSTRING> error. The number of data columns to which you are limited varies, depending on the complexity of the formatting attributes associated with your data columns. Formatting attributes include Width, Alignment, Value Formatting, and even attributes associated with the field definition, such as Internal-to-External Conversion code. If you do not associate any formatting attributes with the data columns, M/PACT allows a maximum of approximately 16 data columns in a report definition. To the extent that you do define formatting attributes, that number decreases. [Open M/SQL Version D.1992-03]

Implementation of SQL

- n The LIKE predicate yields unpredictable results when it is used in an expression that references a field as the <match value> operator (value on the left side of the LIKE predicate), and the field uses the ALPHAUP collation function. [Open M/SQL Version F.7]
- n The INTO clause cannot recover fields from a view into a local array. [Open M/SQL Version D.1992-03]
- n The Interactive Query facility does not support the use of the INTO clause. [Open M/SQL Version D.1992-03]

Open M with SQL for DTM

- n When running Open M with SQL on an Open M for Windows (DTM) system, you should be aware of the following limitations:
 - You must have at least 22 megabytes of free space on your hard disk in order to install Open M with SQL.
 - The Import utility may require large partitions, depending on the size of your import set.
 - The Form Compiler may also require large partitions, depending on the size and complexity of the form(s) you are compiling.

[Open M/SQL Version F.7]

Relational Server

- n You cannot insert, update, or delete dates or rows containing date, time, or multiple line text fields. When inserting a row, you must define its Row ID. [Open M with SQL Version F.9]

Documentation Notes

Set Field Triggers That Target Multi-Line Fields

When defining a Set Field trigger that targets a multi-line field, you can set the trigger to null out all instances of the multi-line field, as follows: [Open M/SQL Version F.6]

- When you specify the field name in the Trigger Definition window, specify just the field name, and do not specify a line number in parentheses.
 - In the Set Field popup window, leave everything blank.
- n You cannot use the Set Field trigger action to set the line counter of a multi-line field. Do not attempt to set the line counter by specifying `FieldName(0)` in the Trigger Definition window. [Open M/SQL Version F.6]

Using Percent Variables in Form Triggers for Multi-Row Forms

- n In multi-row forms, you can only reference row-specific percent variables in Post-Field triggers. The row-specific percent variables are:
- `{%filetype}`
 - `{%presave}`
 - `{%savedata}`
 - `{%return_filetype}`
 - `{%return_presave}`
 - `{%return_savedata}`

These percent variables hold information that is specific to one data row, and therefore, must be used in reference to a specific row.

In single-row forms (where there is only one row), you can also reference these percent variables in Post-Window and Post-Form triggers.

Field Skipping Rows in Multi-Row Forms

- n If a user exits a row in a multi-row form using either the `<DOWN ARROW>` or `<NEXT SCREEN>` keystroke and the field to be landed on is defined to be skipped, the field is skipped as if the user had pressed `<RETURN>` while in it. [Open M/SQL Version E.4]
- n If a row is exited with an UP keystroke (e.g., `<UP ARROW>` or `<PREVIOUS SCREEN>`) and the field to be landed on is to be skipped, the field is skipped as if the user had pressed the `<LEFT ARROW>` key while in it. [Open M/SQL Version E.4]

Available Documentation

Open M with SQL

The documentation set for InterSystems' Open M with SQL relational database product includes the following manuals:

- ⁿ This *Open M with SQL — Version F.11 Release Information*; Revision Date: July, 1996.
- ⁿ *Open M with SQL — Version F.10 Release Information*; Revision Date: November, 1996.
- ⁿ *Open M/SQL Developer Guide* — Version F.6 & F.7; Revision Date: September 11, 1995.
- ⁿ *Open M with SQL Database Administrator's Guide — Version F.9, F.10*; Revision Date: December 9, 1996.
- ⁿ *User Interface Programming Guide — Version F.4*; Revision Date: October 6, 1994.
- ⁿ *Open M with SQL Data Dictionary Guide* — Version F.10; Revision Date: April 2, 1997.
- ⁿ *Open M/SQL M/PACT* (includes *M/PACT Addendum*) — Version B; Revision Date: July 2, 1990.
- ⁿ *Open M with SQL for DTM Installation Guide*
- ⁿ *Open M with SQL for DSM Installation Guide*

Open M Relational Server

If you plan to use Open M Relational Server, you also need the following documentation:

- ⁿ *Open M/SQL Relational Client User Guide* — Version 1.5; Revision Date: April 4, 1995.
- ⁿ *Open M Relational Client Version 2.0 Release Information*; Revision Date: August 23, 1996.
- ⁿ *Open M Relational Client Version 2.1 Release Information*; Revision Date: April 16, 1997.
- ⁿ *Open M/SQL Server Programming Guide* — Version E.3; Revision Date: May 5, 1993. *Available on request.*
- ⁿ *Open M Relational Server Manager's Guide* — Version F.10; Revision Date: December 9, 1996.

Converting Your Applications to the New Global Structure for Routines

New Global Structure for Routines

Open M with SQL routine code is now stored in five separate globals. Previously, all routines were stored exclusively in the ^ROUTINE global¹. This was done to distribute routine source code across directories and/or systems. When you upgrade to Open M with SQL Version F.11, you must convert your existing Open M with SQL routines to the new global structure.

Open M routines can have different extensions and version numbers. The extension describes the type of routine file, and the version differentiates between multiple copies of the same routine. A routine can have the following extensions:

- .MAC — Macro Source Routine
- .INT — Intermediate Source Routine
- .INC — Include File
- .OBJ — Compiled Object Code

Under the new global structure for routine storage, the intermediate code can be in one directory, the macro source code in another and etc. The five globals used for Open M with SQL routine storage are:

- ^ROUTINE
- ^rINC
- ^rINCSAVE
- ^rMAC
- ^rMACSAVE

1. For Open M with SQL Version F.10, the global routines were called ^ROUTINE, ^mINC, ^mINCSAVE, ^mMAC, and ^mMACSAVE.

Routine Storage Format

Table A-1 shows how routines are mapped to the new structure:

Table A-1: Routine Storage Format

Routine Extension	Old Global Format	New Global Format
.MAC	^ROUTINE(0,"MAC",Version, Routine_Name,0)=Date of last edit 0,0)=Number of code lines 0,n)=Line of code	^rMAC(Routine_Name,0)= Date of last edit 0,0)=Number of code lines 0,n)=Line of code
		for backup versions ¹ : ^rMACSAVE(Routine_Name,Version)= Date of last edit Version,0)=Number of code lines Version,n)=Line of code
.INC	^ROUTINE(0,"INC",Version, Routine_Name)	^rINC(Routine_Name,0)= Date of last edit 0,0)=Number of code lines 0,n)=Line of code
		for backup versions: ^rINCSAVE(Routine_Name,Version)= Date of last edit Version,0)=Number of code lines Version,n)=Line of code
.INT	^ROUTINE(Routine_Name,0)= Date of last edit 0,0)=Number of code lines 0,n)=Line of code	Same as old format

1. The highest version number reflects the most recent backup.

Parameter	Meaning
<i>Date of last edit</i>	Date that the routine was last edited. Stored in the M standard \$H format.
<i>Number of code lines</i>	The total number of code lines in the routine.
<i>Lines of code</i>	The individual code and comment lines that make up the routine.
<i>Version</i>	The version number of the routine. Zero (0) is the current version. Highest version number is the latest backup.
<i>Routine_Name</i>	Name of the routine stored in this global node

ROUTINE Global Conversion Utility

When you upgrade to Open M with SQL Version F.11, you must convert your existing Open M with SQL routines to the new global structure. Follow the appropriate procedure for your system as described below:

For Open M NextGen 2.0 Systems:

Any existing routine global names are converted as part of the system conversion routine, %SYSCONV. See your *Open M NextGen Installation Guide* for information on system conversion.

For Open M with SQL 6.x, Open M with SQL for DSM, and Open M with SQL for DTM Systems:

The %urconv utility converts the ^ROUTINE global to the new global storage format. To convert your existing routines to the new global structure:

```
>d all^%urconv
```

Note: The %urconv utility must be run on a per directory basis for **all** application directories. It should not be run in the manager's directory or the Open M with SQL common directory.

The entry points to the ^%urconv routine are listed and described in the table below.

Table A-2: Entry Points into %urconv

Entry Point	Action
d all^%urconv	Converts all .MAC and .INC files in the current directory to the new structure.
d ^%urconv	Converts selected routine(s) to the new structure. Enter routine specifications or ? at the Routine prompt.
d rtnset^%urconv	Converts your predefined routine sets to be compatible with the new format. These routine sets are stored in the global ^mutil("rset").

Setting the Number of Backup Versions

You can set the maximum number of versions maintained for Macro Source routines (.MAC) and Include files (.INC). Intermediate source routines (.INT) can only have one copy at a time. Run the %urverma routine to set this version limit:

```
>d ^%urverma  
Number of versions to keep for .MAC:  4 =>  
Number of versions to keep for .INC:  4 =>
```

Note: The version limit should be a number greater than 0.

InterSystems SQL DDL Grammar

Conventions used in DDL Grammar

This section observes the following typographic conventions for InterSystems SQL DDL Grammar:

Convention Description	Example
Angle braces surround elements of the SQL command.	<column name list> :
Square brackets indicate optional parameters.	[<comma> <column name>]
An ellipsis indicates repeating the element enclosed in curly braces as necessary.	{ <comma> <column name> }...
InterSystems extensions to the SQL standard are underlined.	<u><table description specification></u>

Create Table Command

<create table statement> ::= CREATE TABLE <table name> <table element list>

<table element list> ::= <left paren> <table element> [{ <comma> <table element> }...] <right paren>

<table element> ::= <column definition> | <table constraint definition> |

<table description specification> | <table file specification> |

<table numrows specification> | <table routine specification>

<column definition> ::= <column name> <data type> [<collate clause>] [<multi-line clause>]

[<description clause>] [<default clause>] [<column constraint definition>...]

<data type> ::= <character string type> | <numeric type> | <name type> | <date type> | <time type> | <mssql number type> | <yesno type>

<character string type> ::=

CHARACTER [<left paren> <length> <right paren>]

| CHAR [<left paren> <length> <right paren>]

| CHARACTER VARYING <left paren> <length> <right paren>

| CHAR VARYING <left paren> <length> <right paren>

| VARCHAR <left paren> <length> <right paren>

<numeric type> ::= <exact numeric type> | <approximate numeric type>

<exact numeric type> ::=

NUMERIC [<left paren><precision> [<comma><scale>] <right paren>]

| DECIMAL [<left paren><precision> [<comma><scale>] <right paren>]

| DEC [<left paren> <precision> [<comma> <scale>] <right paren>]

| INTEGER

| INT

| SMALLINT

<precision> ::= <unsigned integer>

<scale> ::= <unsigned integer>

<approximate numeric type> ::=

FLOAT [<left paren> <precision> <right paren>]

| REAL

| DOUBLE PRECISION

<name type> ::= %NAME [<left paren> <length> <right paren>]

<date type> ::= DATE [<date parameter list>]

<date parameter list> ::= <date named param. list> | <date ordered param. list>

<date named parameter list> ::=

<left paren> <date parameter> [{<comma> <date parameter>} ...] <right paren>

<date parameter> ::= DATEFORMAT <equal> <date format>

| FIRSTDATE <equal> <date literal>

| LASTDATE <equal> <date literal>

| DATEFULLYEAR <equal> <yesno literal>

| DATEDELIMITER <equal> <character literal>

<equal> ::= '='

<date format> ::= <unsigned integer>

(== Equal to a valid M/SQL Date Format)

<date literal> == External format M/SQL date literal

<yesno literal> ::= 'Yes' | 'No'

<date ordered parameter list> ::=

<left paren> <date format> <comma> <first date> <comma> <last date>
<comma> <display full year> <comma> <display delimiter> <right paren>

<first date> ::= <date literal>

<last date> ::= <date literal>

<display full year> ::= <yesno literal>

<display delimiter> ::= <character literal>

<time type> ::= TIME [<time parameter list>]

<time parameter list> ::= <time named parameter list> | <time ordered parameter list>

<time named parameter list> ::= <time parameter> [<comma> <time parameter> ...]

<time parameter> ::= TIMEFORMAT <equal> <time format>

| FIRSTTIME <equal> <time literal>

| LASTTIME <equal> <time literal>

<time format> ::= <unsigned integer> (== Equal to
a valid M/SQL Time Format number)

<time literal> == External format M/SQL time literal

<time ordered parameter list> ::= <left paren> <time format>
<comma> <first time> <comma> <last time> <right paren>

<first time> ::= <time literal>

<last time> ::= <time literal>

<msql number type> ::= %NUMBER [<number parameter list>]

<number parameter list> ::= <number named parameter list> |
<number ordered parameter list>

<number named parameter list> ::=
<number parameter> [<comma> <number parameter> ...]

<number parameter> ::= NUMFORMAT <equal> <number format>
| NUMMIN <equal> <number literal>
| NUMMAX <equal> <number literal>
| NUMDECS <equal> <number literal>
| NUMLEADPUNC <equal> <character literal>
| NUMMINUSFMT <equal> <number minus literal>
| NUMUSEDMATH <equal> <yesno literal>

<number ordered parameter list> ::=
<left paren><number format><comma> <number min> <comma>
<number max> <comma> <number decimals> <comma>
<leading punctuation> <comma> <number minus literal>
<comma><used in arithmetic> <right paren>

<number format> ::= <unsigned integer>

<number literal> == External format M/SQL number literal.

<number minus literal> ::= '-X' | 'X-' | '(X)'

<number min> ::= <number literal>

<number max> ::= <number literal>

<number decimals> ::= <number literal>

<used in arithmetic> ::= <yesno literal>

<yesno type> ::= %YESNO

<collate clause> ::= %ALPHAUP | %UPPER | %EXACT

<multi-line clause> ::= %MULTILINE

<description clause> ::= %DESCRIPTION <literal>

<default clause> ::= DEFAULT <default option>

<default option> ::= NULL | <constant> | <mcode expression>

<mcode expression> ::= %MCODE <m expression>
<m expression> ::= M code to be executed enclosed in quotes.
Internal quotes must be doubled.

```

<column constraint definition> ::= [ <constraint name definition> ] <column constraint>

<constraint name definition> ::= CONSTRAINT <constraint name>

<constraint name> ::= <identifier>

<column constraint> ::= NOT NULL | <unique specification> | <references specification>

<unique specification> ::= UNIQUE | PRIMARY KEY

<references specification> ::= REFERENCES <referenced tables and columns>
                                [ MATCH<match type> ]
                                [<referential triggered action>]

<referenced tables and columns> ::= <table name>
                                [<left paren> <reference column list><right paren>]

<reference column list> ::= <column name list>

<column name list> ::= <column name>[(<comma><column name>)...]

<match type> ::= FULL | PARTIAL

<referential trigger action> ::= <update rule> [ <delete rule> ] |
                                <delete rule> [<update rule>]

<update rule> ::= ON UPDATE <referential action>

<delete rule> ::= ON DELETE <referential action>

<referential action> ::= CASCADE | SET NULL |
                        SET DEFAULT | NO ACTION

<table constraint definition> ::= [ <constraint name definition> ] <table constraint>

<constraint name definition> ::= CONSTRAINT <constraint name>

<constraint name> ::= <identifier>

<table constraint> ::= <unique constraint definition> | <referential constraint definition>

<unique constraint definition> ::= <unique specification> <left paren> -
                                <unique column list><right paren>

<unique column list> ::= <column name list>

<referential constraint definition> ::= FOREIGN KEY <left paren>
                                <referencing columns> <right paren> <references specification>

<referencing columns> ::= <reference column list>

```

<table description specification> ::= %DESCRIPTION <constant>

<table file specification> ::= %FILE <global specification>

<global specification> ::= character string literal equal to a valid M global reference with
no subscripts

If '^' is omitted, it is automatically added.

Not validated until DDL Statement run-time

<table numrows specification> ::= %NUMROWS <integer>

<table routine specification> ::= %ROUTINE <routine specification>

<routine specification> ::= character string literal equal to a valid M routine name.

Not validated until DDL Statement run-time

<left paren> ::= (

<right paren> ::=)

<comma> ::= ,

Examples 1. &sql(CREATE TABLE DDLTest2 (

```
C1      CHARACTER
        %DESCRIPTION 'C1 Field - Text of length 1',
C2      CHAR(20)
        %DESCRIPTION 'C2 Field - Text of length 20',
C3      CHARACTER VARYING (10)
        %DESCRIPTION 'C3 Field - Text of length 10',
C4      CHAR VARYING(11)
        %DESCRIPTION 'C4 Field - Text of length 11',
C5      VARCHAR (5)
        %DESCRIPTION 'C5 Field - Text of length 5',
N1      NUMERIC
        %DESCRIPTION 'N1 Field - Number',
N2      NUMERIC(3)
        %DESCRIPTION 'N2 Field - # of precision 3',
N3      NUMERIC(3,2)
        %DESCRIPTION 'N3 Fld - # of prec 3, scal 2',
N4      DECIMAL
        %DESCRIPTION 'N4 Fld - Number',
N5      DECIMAL(5)
        %DESCRIPTION 'N5 Fld - # of prec 5',
N6      DECIMAL(6,2)
        %DESCRIPTION 'N6 Fld - # of prec 6, scal 2',
N7      DEC
        %DESCRIPTION 'N7 Fld - Number',
N8      DEC(5)
        %DESCRIPTION 'N8 Fld - # of prec 5',
N9      DEC(6,2)
        %DESCRIPTION 'N9 Fld - # of prec 6, scal 2',
N10     INTEGER
        %DESCRIPTION 'N10 Fld - Number, 0 decimals',
N11     INT
        %DESCRIPTION 'N11 Fld - Number, 0 decimals',
```

```

N12          SMALLINT

N13          %DESCRIPTION 'N12 Fld - Number, 0 decimals',
            FLOAT

N14          %DESCRIPTION 'N13 Fld - Number, approx',
            FLOAT(5)

N15          %DESCRIPTION 'N14 Fld - #, approx, prec 5',
            REAL

N16          %DESCRIPTION 'N15 Fld - Number, approx',
            DOUBLE PRECISION

N17          %DESCRIPTION 'N16 Fld - Number, approx',
            %NUMBER

N18          %DESCRIPTION 'N17 Fld - Number',
            %NUMBER (NUMFORMAT = 1,
                    NUMMIN = -1000,
                    NUMMAX = 1000,
                    NUMDECS = 2,
                    NUMLEADPUNC = '$',
                    NUMMINUSFMT = '-X',
                    NUMUSEDMATH = 'Yes')

N19          %DESCRIPTION 'N18 Fld - Number, named param',
            %NUMBER ('1', '-1000', '1000',
                    '2', '$', '-X', 'Yes')

NA1          %DESCRIPTION 'N19 Fld - Num, ordered param',
            %NAME(30)

D1          %DESCRIPTION 'NA1 Field - Name',
            DATE (DATEFORMAT = 1,
                FIRSTDATE = '09/10/96',
                LASTDATE = '12/31/99',
                FULLYEAR = 'Yes',
                DELIMITER = '*')

D2          %DESCRIPTION 'D1 Field - Date, named param',
            DATE(1, '1/1/90', '1/1/99', 'No', '/')

T1          %DESCRIPTION 'D2 Fld - Date, ordered param',
            TIME (TIMEFORMAT = 1,
                FIRSTTIME = '12:00AM',
                LASTTIME = '12:0PM')

T2          %DESCRIPTION 'T1 Field - Time, named param',
            TIME(1, '12:00PM', '12:00AM')

Y1          %DESCRIPTION 'T2 Fld - Time, ordered param',
            %YESNO

MUL1        %DESCRIPTION 'Y1 Field - Yes/No',
            CHAR(10)

REQ1        %MULTILINE
            %DESCRIPTION 'Multiline Text field',
            CHAR(15)

REQ2        %DESCRIPTION 'Required text field with def'
            DEFAULT '<new value>'
            NOT NULL,
            CHAR(15)

COL1        %DESCRIPTION 'Req text field with M def'
            DEFAULT %MCODE '$$next(^myglo)'
            NOT NULL,
            CHAR(10)

COL2        %ALPHAUP
            %DESCRIPTION 'Text with ALPHAUP collation',
            CHAR(10)

COL3        %UPPER
            %DESCRIPTION 'Text with UPPER collation',
            CHAR(10)

%FILE        %EXACT
            %DESCRIPTION 'Text with EXACT collation',
            '^ddltest2',
            %DESCRIPTION 'Test DDL #2',
            %NUMROWS 50,
            %ROUTINE 'ddltest2'))

```

2.

```
&sql(CREATE TABLE DDLState (  
  
    StateName      CHARACTER(30)  
                  %ALPHAUP  
                  %DESCRIPTION 'State Name'  
                  CONSTRAINT nn NOT NULL  
                  CONSTRAINT uu UNIQUE  
                  CONSTRAINT pk PRIMARY KEY,  
    Abbreviation   CHARACTER(2)  
                  %UPPER  
                  %DESCRIPTION 'State Abbreviation'  
                  UNIQUE  
                  NOT NULL,  
    Test           CHARACTER(10)  
                  DEFAULT NULL,  
    T1             CHARACTER(30) UNIQUE,  
    T2             CHARACTER(30) UNIQUE,  
    %FILE          '^state',  
    %DESCRIPTION   "U.S. States",  
    %NUMROWS       50,  
    %ROUTINE       'states',  
    CONSTRAINT uniq UNIQUE (StateName),  
    CONSTRAINT pkey PRIMARY KEY (StateName),  
    CONSTRAINT tu   UNIQUE(T1,T2)))
```
3.

```
&sql(CREATE TABLE DDLPerson (  
  
    %FILE          '^person',  
    %NUMROWS       1000000,  
    %ROUTINE       'people',  
    %DESCRIPTION   'People I know',  
    Name           CHARACTER (30) NOT NULL,  
    Address        CHARACTER (50),  
    PhoneNumber    CHARACTER (20),  
    State          CHARACTER (30) REFERENCES DDLState (StateName)  
                  MATCH FULL ON UPDATE SET DEFAULT  
                  ON DELETE SET DEFAULT,  
    S1             CHARACTER (30) UNIQUE,  
    S2             CHARACTER (30) UNIQUE,  
    CONSTRAINT c1 FOREIGN KEY (S1, S2)  
                  REFERENCES DDLState (T1, T2)))
```

Alter Table Command

<alter table statement> ::= ALTER TABLE <table name> <alter table action>

<alter table action> ::= <add column definition> | <alter column definition> | <drop column definition>

<add column definition> ::= ADD [COLUMN] <column definition>

<column definition> ::= (See CREATE TABLE BNF)

<alter column definition> :: ALTER [COLUMN] <column name> <alter column action>

<alter column action> ::= <set column default clause> | <drop column default clause>

<set column default clause> ::= SET <default clause>

<default clause> ::= DEFAULT <default option>

<default option> ::= <literal> | NULL | <mcode expression>

<mcode expression> ::= %MCODE <m expression>

<m expression> ::= M code to be executed enclosed in quotes. Internal quotes must be doubled.

<drop column default clause> ::= DROP DEFAULT

**<drop column definition> ::= DROP [COLUMN] <column name> [<drop behavior>]
[<delete data behavior>]**

<drop behavior> ::= CASCADE | RESTRICT

<delete data behavior> ::= %DELDATA | %NODELDATA

%DELDATA = Delete the column's data before dropping

%NODELDATA = Do not delete the column's data before dropping

**If <delete data behavior> is not specified, the system wide default setting is used.
This is defined in the System Configuration Settings, SQL/Table Options window.**

Alter Table Notes:

- n After each Alter Table statement, the table is compiled silently in the foreground.
- n A user may only execute an ALTER TABLE DDL statement if the user has ALTER privilege for the Base Table.
- n A user may only execute an ALTER TABLE DDL statement if the user has Data Dictionary Access specified in the user's definition.
- n A Column which is, or is part of, the table's Primary Key may not be deleted with the ALTER TABLE DROPCOLUMN statement. An SQL error 327 occurs.

- n We do not support ALTER TABLE ADD <table constraint definition>.
- n We do not support ALTER TABLE DROP CONSTRAINT <constraint name>...

Examples

1. Create a new field in the DDLState table called Text:

```
&sql(ALTER TABLE DDLState ADD COLUMN Text CHARACTER (20))
```

2. Delete the Text field from the DDLState table:

```
&sql(ALTER TABLE DDLState DROP COLUMN Text)
```

3. Change the default value of the field DDLTest.Number2 to 1:

```
&sql(ALTER TABLE DDLTest  
      ALTER COLUMN Number2 SET DEFAULT '1')
```

4. Change the default value of the field DDLTest.Number2 to '':

```
&sql(ALTER TABLE DDLTest  
      ALTER COLUMN Number2 SET DEFAULT NULL)
```

5. More examples:

```
&sql(ALTER TABLE DDLTest DROP F3 CASCADE)  
&sql(ALTER TABLE DDLTest DROP F3 RESTRICT)  
&sql(ALTER TABLE DDLTest DROP COLUMN F3)  
&sql(ALTER TABLE DDLTest DROP COLUMN F3 CASCADE)  
&sql(ALTER TABLE DDLTest DROP COLUMN F3 RESTRICT)  
&sql(ALTER TABLE DDLTest DROP F3 %DELDATA)  
&sql(ALTER TABLE DDLTest DROP F3 CASCADE %DELDATA)  
&sql(ALTER TABLE DDLTest DROP F3 RESTRICT %DELDATA)  
&sql(ALTER TABLE DDLTest DROP COLUMN F3 %DELDATA)  
&sql(ALTER TABLE DDLTest DROP COLUMN F3 CASCADE %DELDATA)  
&sql(ALTER TABLE DDLTest DROP COLUMN F3 RESTRICT %DELDATA)  
&sql(ALTER TABLE DDLTest DROP F3 %NODELDATA)  
&sql(ALTER TABLE DDLTest DROP F3 CASCADE %NODELDATA)  
&sql(ALTER TABLE DDLTest DROP F3 RESTRICT %NODELDATA)  
&sql(ALTER TABLE DDLTest DROP COLUMN F3 %NODELDATA)  
&sql(ALTER TABLE DDLTest DROP COLUMN F3 CASCADE %NODELDATA)  
&sql(ALTER TABLE DDLTest DROP COLUMN F3 RESTRICT %NODELDATA)
```


Drop Table Command

<drop table statement> ::= DROP TABLE <table name> [<drop behavior>] [<delete data behavior>]

<drop behavior> ::= CASCADE | RESTRICT

CASCADE = Delete all dependents of the Table

RESTRICT = Do not delete if table has dependents

<delete data behavior> ::= %DELDATA | %NODELDATA

%DELDATA == Delete the table's data before dropping

%NODELDATA == Do not delete the table's data before dropping

If <delete data behavior> is not specified, the system wide default setting is used.
This is defined in the System Configuration Settings, SQL/Table Options window.

Drop Table Notes:

- n A base table's dependents are views which include the table and other base tables which have designated reference fields which point to it.
- n A user may only execute a DROP TABLE DDL statement if that user has ALTER privilege for the base table.
- n A user may only execute an DROP TABLE DDL statement if the user has Data Dictionary Access specified in the user's definition.

Examples 1. The following example deletes the table DDLTest and its data:

```
&sql(DROP TABLE DDLTest %DELDATA)
```

2. The following example deletes the table DDLTest, but does not delete the table's data:

```
&sql(DROP TABLE DDLTest %NODELDATA)
```

3. More examples:

```
&sql(DROP TABLE DDLPerson)
&sql(DROP TABLE DDLState CASCADE)
&sql(DROP TABLE DDLState RESTRICT)
&sql(DROP TABLE DDLState CASCADE %DELDATA)
&sql(DROP TABLE DDLState CASCADE %NODELDATA)
&sql(DROP TABLE DDLState RESTRICT %DELDATA)
&sql(DROP TABLE DDLState RESTRICT %NODELDATA)
&sql(DROP TABLE DDLState %DELDATA)
&sql(DROP TABLE DDLState %NODELDATA)
```

Create Index Command

<create index statement> ::= CREATE INDEX <index name> ON TABLE <table name><field element list>

<field element list> ::= <left paren> <field element> [<comma> <field element> ...] <right paren>

<field element> ::= <field name> [<ordering specification>]

<ordering specification> ::= ASC | DESC

Create Index Notes:

- n <index name> is unique within a database.
- n <ordering specification> is parsed, but not used.
- n Index is populated with any existing data automatically.
- n CREATE INDEX is not part of the SQL standard.
- n A user may only execute a CREATE INDEX DDL statement if the user has ALTER privilege for the Base Table.
- n A user may only execute an CREATE INDEX DDL statement if the user has Data Dictionary Access specified in the user's definition.

Examples

1. The following example creates an index named StateName on the State table indexing the field StateAbbrev.

```
&sql(CREATE INDEX StateName ON TABLE State (StateAbbrev))
```

2. The following index creates an index named TempIndex on the temp.f1 and Temp.f2 fields. The ASC (or DESC) clause is ignored. The parser supports it for compatibility with 3rd party tools.

```
&sql(CREATE INDEX TempIndex ON TABLE Temp (f1 ASC, f2 DESC))
```

Drop Index Command

<drop index statement> ::= DROP INDEX <index name> [<table clause>]

<table clause> ::= ON TABLE <table name>

Drop Index Notes:

- n Currently, the indexes' data is not deleted
- n DROP INDEX is not part of the SQL standard
- n A user may only execute an DROP INDEX statement if the user has ALTER privilege for the Base Table.
- n A user may only execute an DROP INDEX DDL statement if the user has Data Dictionary Access specified in the user's definition.

Examples 1. The following example deletes the index StateName:

```
&sql(DROP INDEX StateName)
```

2. The following index also deletes the index StateName. The ON TABLE State clause is optional and ignored since index names are unique within a database. It is allowed for compatibility with 3rd party tools.

```
&sql(DROP INDEX StateName ON TABLE State)
```

DDL Error Messages

New SQLCODE error messages are included in Open M with SQL version F.11 for the implementation of DDL.

Table B-1: SQLCODE Errors for DDL

SQLCODE	Error Message
301	Cannot start DDL transaction
302	Unknown datatype
303	No DDL transaction started
304	Another DDL transaction already in progress
305	Cannot create column for non-existent table
306	Column with this name already exists
307	Primary key already defined for this table
308	Primary key definition has no columns defined
309	Primary key definition includes invalid column(s)
310	Foreign key references non-existent table
311	Local key and referenced key do not have same column count
312	Referenced key has no columns defined
313	Referenced key includes invalid column(s)
314	Foreign key references non-unique key/column collection
315	Cannot create key for non-existent table
316	Key with this name already defined for this table
317	Local key references invalid key in foreign table
318	Unable to create candidate key in referenced table
319	Referenced table has no primary key defined
320	User access denied
321	No such column defined
322	Cannot alter table with customized structure
323	Foreign key includes no columns from local table
324	Index with this name already defined for this table
325	Index on specified columns already defined
326	Invalid columns referenced in index specification
327	Cannot DELETE a column which is part of the primary key

Table B-1: SQLCODE Errors for DDL

SQLCODE	Error Message
328	Invalid Datatype Parameter name specified
329	Invalid key type specified
330	Cannot start DDL transaction due to unknown user ID
331	Cannot drop key from non-existent table
332	No such constraint defined for this table
333	No such index defined
334	Cannot delete table because it has dependents
335	No such table defined
336	Invalid trigger location specified
337	Invalid row selection location specified
338	Cannot specify existing table as child
339	Table is not child of specified parent
340	DDL nested within pre-existing transaction - not rolled back
341	No such map defined.

