

Cincom

AD/ADVANTAGE

MANTIS DB2 Programming
OS/390, VSE/ESA

P39-5028-00



AD/Advantage[®] MANTIS DB2 Programming OS/390, VSE/ESA

Publication Number P39-5028-00

© 1988–1998, 2001 Cincom Systems, Inc.
All rights reserved

This document contains unpublished, confidential, and proprietary information of Cincom. No disclosure or use of any portion of the contents of these materials may be made without the express written consent of Cincom.

The following are trademarks, registered trademarks, or service marks of Cincom Systems, Inc.:

AD/Advantage [®]	iD CinDoc [™]	MANTIS [®]
C+A-RE [™]	iD CinDoc Web [™]	Socrates [®]
CINCOM [®]	iD Consulting [™]	Socrates [®] XML
Cincom Encompass [®]	iD Correspondence [™]	SPECTRA [™]
Cincom Smalltalk [™]	iD Correspondence Express [™]	SUPRA [®]
Cincom SupportWeb [®]	iD Environment [™]	SUPRA [®] Server
CINCOM SYSTEMS [®]	iD Solutions [™]	Visual Smalltalk [®]
	intelligent Document Solutions [™]	VisualWorks [®]
gOOj [™]	Intermax [™]	

All other trademarks are trademarks or registered trademarks of:

Acucobol, Inc.	Micro Focus, Inc.
AT&T	Microsoft Corporation
Compaq Computer Corporation	Systems Center, Inc.
Data General Corporation	TechGnosis International, Inc.
Gupta Technologies, Inc.	The Open Group
International Business Machines Corporation	UNIX System Laboratories, Inc.
JSB Computer Systems Ltd.	

or of their respective companies.

Cincom Systems, Inc.
55 Merchant Street
Cincinnati, OH 45246-3732
U.S.A.

PHONE: (513) 612-2300
FAX: (513) 612-2000
WORLD WIDE WEB: <http://www.cincom.com>

Attention:

Some Cincom products, programs, or services referred to in this publication may not be available in all countries in which Cincom does business. Additionally, some Cincom products, programs, or services may not be available for all operating systems or all product releases. Contact your Cincom representative to be certain the items are available to you.

Release information for this manual

AD/Advantage MANTIS DB2 Programming, OS/390, VSE/ESA, P39-5028-00, is dated October 30, 2001. This document supports Release 5.5.01 of MANTIS SQL Support.

We welcome your comments

We encourage critiques concerning the technical content and organization of this manual. Please take the [survey](#) provided with the online documentation at your convenience.

Cincom Technical Support for AD/Advantage

All customers

Web: <http://supportweb.cincom.com>

U. S. A. customers

Phone: 1-800-727-3525

FAX: (513) 612-2000

Attn: AD/Advantage Support

Mail:

Cincom Systems, Inc.

Attn: AD/Advantage Support

55 Merchant Street

Cincinnati, OH 45246-3732

U. S. A.

Customers outside U. S. A.

All:

Visit the support links at

<http://www.cincom.com> to find

contact information for your nearest

Customer Service Center.



Contents

About this book	ix
Using this document.....	ix
Document organization.....	ix
Conventions.....	xi
MANTIS documentation series.....	xiv
Educational material	xv
Overview of MANTIS SQL support	17
Embedding SQL statements in MANTIS programs.....	18
Static and dynamic SQL statements	19
MANTIS SQL support execution modes	20
Dynamic execution mode (DB2 for VSE and VM—formerly SQL/DS—and DB2 for OS/390)	20
Static execution mode (DB2 for OS/390)	20
Extended dynamic execution mode (DB2 for VSE and VM—formerly SQL/DS)	21
Embedding SQL statements in MANTIS programs	23
Rules for embedding SQL statements in MANTIS programs.....	23
Coding host variables in SQL statements	27
Coding indicator variables in SQL statements.....	31
Converting data between MANTIS SQL support and the SQL database.....	32
Specifying SQL data types in host variables	34
Programming considerations	37
SQL statement limits	39
Scope of SQL cursors and statements.....	40
SQL WHENEVER statement.....	40
Using SQL WHENEVER as a declarative statement	44
Scope of the WHENEVER statement	45

SQLCA in MANTIS SQL support	45
SQLCA statement syntax.....	45
SQLCA function syntax	47
SQLCA elements	48
SQL COMMIT WORK and ROLLBACK WORK statements	50
Binding with the High-Performance Option (HPO)	51
Running a program from a line number	51
Error messages.....	52
Maximum number of host variables.....	54
Using ENTRY statement parameters as host variables	54

Dynamic SQL statements 55

SQLDA statement and function	57
Allocate an SQLDA	59
Deallocate an SQLDA.....	60
Set SQLDA header information	61
Move data into an SQLDA repeating group	64
Read header elements.....	67
Move data from an SQLDA repeating group into a MANTIS program.....	69
SQL statements larger than 254 characters	72

Preparing MANTIS SQL support programs for static execution mode (DB2 for OS/390) 75

Static execution mode vs. dynamic execution mode	75
Static execution mode.....	75
Dynamic execution mode.....	75
Preparing your programs to run in static execution mode	76
Performance and programming considerations.....	78
Determining host variable data types.....	78
SQL indicator variables	78
Declaring SQL cursors multiple times.....	79
ENTRY statement parameters as host variables.....	80
Multiple row result sets with SELECT	81
Binding on the target system.....	82
Preparing a program to run in static mode.....	82
SQL binding the MANTIS program	82
Generating the SQL support source module	93
Creating the SQL support load module.....	97
Creating the DB2 application plan and granting execution authority	98
Making SQL support load modules available to MANTIS	99
Checking the consistency of a bound program.....	104
Unbinding SQL bound programs	107
Maintaining SQL bind information	110
Running SQL support programs with batch MANTIS	112

MANTIS SQL support programs for extended dynamic execution mode (DB2 for VSE and VM—formerly SQL/DS)	113
Performance and programming considerations	115
Determining host variable data types	115
Multiple row result sets with SELECT	116
Binding on the target system	117
Preparing a program to run in extended dynamic execution mode	117
SQL binding a MANTIS program online	118
SQL binding a MANTIS program with batch MANTIS	125
Checking the results of an SQL bind	127
Checking the consistency of a bound program	128
Unbinding SQL bound programs	131
Sample MANTIS SQL support programs	135
Insert program using static SQL statements	136
Insert program using dynamic SQL statements	138
Update program using static SQL statements	140
Update program using dynamic SQL statements	141
Select program using static SQL statements	142
Select program using dynamic SQL statements	143
Delete program using static SQL statements	144
Delete program using dynamic SQL statements	145
Column select program using dynamic SQL statements	146
Hold cursors across a COMMIT	147
Features not supported	149
Comparing SQL in MANTIS SQL support to SQL in COBOL	151
SQL keywords	155
Index	157

About this book

Using this document

MANTIS is an application development system that consists of design facilities (e.g., screens and files) and a programming language. This manual provides information on how to use MANTIS SQL Support to create MANTIS applications that access databases with SQL, including programming and execution information.

Document organization

The information in this manual is organized as follows:

Chapter 1—Overview of MANTIS SQL support

Provides an overview of SQL support in MANTIS.

Chapter 2—Embedding SQL statements in MANTIS programs

Describes how to code host and indicator variables in SQL statements, convert data between MANTIS support and the SQL database, and specify SQL data types in host variables.

Chapter 3—Programming considerations

Provides limits and specific considerations for MANTIS SQL support.

Chapter 4—Dynamic SQL statements

Describes the differences between dynamic SQL statements in MANTIS SQL Support and dynamic SQL statements in SQL in COBOL.

Chapter 5—Preparing MANTIS SQL support programs for static execution mode (DB2 for OS/390)

Describes the steps required to prepare your programs to run in static execution mode.

Chapter 6—MANTIS SQL support programs for extended dynamic execution mode (DB2 for VSE and VM)

Describes the differences in the procedure used by MANTIS to create the Access Module and that used by SQL in COBOL.

Extended dynamic execution mode allows the SQL statements in MANTIS SQL programs to be compiled into a DB2 for VSE and VM Access Module just as they are for SQL in COBOL.

Appendix A—Sample MANTIS SQL support programs

Contains sample MANTIS SQL support programs using static and dynamic SQL statements.

Appendix B—Features not supported

Lists features not supported by MANTIS SQL.

Appendix C—Comparing SQL in MANTIS SQL support to SQL in COBOL

Provides general considerations applied to SQL in MANTIS SQL Support as compared to SQL in COBOL.

Appendix D—SQL keywords

Lists the SQL keywords used by MANTIS SQL support.

Index

Convention	Description	Example
Braces { }	<p>Indicate selection of parameters. (Do not attempt to enter braces or to stack parameters.) Braces surrounding stacked items represent alternatives, one of which you must select.</p> <p>The example indicates that you must enter FIRST, LAST, or a value for <i>begin</i>.</p>	<pre>{ FIRST begin LAST }</pre>
<p><u>Underlining</u> (In syntax)</p>	<p>Indicates the default value supplied when you omit a parameter.</p> <p>The example indicates that if you do not specify ON, OFF, or a row and column destination, the system defaults to ON.</p> <p>Underlining also indicates an allowable abbreviation or the shortest truncation allowed.</p> <p>The example indicates that you can enter either PRO or PROTECTED.</p>	<pre>SCROLL [ON OFF [row] [, col]]</pre> <p><u>PROTECTED</u></p>
Ellipsis points...	<p>Indicate that the preceding item can be repeated.</p> <p>The example indicates that you can enter (A), (A,B), (A,B,C), or some other argument in the same pattern.</p>	<pre>(argument , ...)</pre>

Convention	Description	Example
UPPERCASE	<p>Indicates MANTIS reserved words. You must enter them exactly as they appear.</p> <p>The example indicates that you must enter CONVERSE exactly as it appears.</p>	CONVERSE <i>name</i>
<i>Italics</i>	<p>Indicate variables you replace with a value, a column name, a file name, and so on.</p> <p>The example indicates that you can supply a name for the program.</p>	COMPOSE [<i>program-name</i>]
Punctuation marks	<p>Indicate required syntax that you must code exactly as presented.</p> <p>() parentheses . period , comma : colon ; semicolon ' single quotation mark " " double quotation marks</p>	[LET] _v $\begin{matrix} (i) \\ (i, j) \end{matrix}$ [ROUNDED(<i>n</i>)] = <i>e1</i> [, <i>e2</i> , <i>e3</i> ...]

MANTIS documentation series

MANTIS is an application development system designed to increase productivity in all areas of application development, from initial design through production and maintenance. MANTIS is part of AD/Advantage, which offers additional tools for application development. Listed below are the manuals offered with MANTIS in the IBM® mainframe environment, organized by task. You may not have all the manuals listed here.

MASTER User tasks

- ◆ *MANTIS Installation, Startup, and Configuration, MVS/ESA, OS/390, P39-5018*
- ◆ *MANTIS Installation, Startup, and Configuration, VSE/ESA, P39-5019*
- ◆ *MANTIS Administration, OS/390, VSE/ESA, P39-5005*
- ◆ *MANTIS Messages and Codes, OS/390, VSE/ESA, P39-5004**
- ◆ *MANTIS Administration Tutorial, OS/390, VSE/ESA, P39-5027*
- ◆ *MANTIS XREF Administration, OS/390, VSE/ESA, P39-0012*

General use

- ◆ *MANTIS Quick Reference, OS/390, VSE/ESA, P39-5003*
- ◆ *MANTIS Facilities, OS/390, VSE/ESA, P39-5001*
- ◆ *MANTIS Language, OS/390, VSE/ESA, P39-5002*
- ◆ *MANTIS Program Design and Editing, OS/390, VSE/ESA, P39-5013*
- ◆ *MANTIS Messages and Codes, OS/390, VSE/ESA, P39-5004**
- ◆ *AD/Advantage Programming, P39-7001*
- ◆ *MANTIS DB2 Programming, OS/390, VSE/ESA, P39-5028*

- ◆ *MANTIS SUPRA SQL Programming, OS/390, VSE/ESA*, P39-3105
- ◆ *MANTIS XREF, OS/390, VSE/ESA, OpenVMS*, P39-0011
- ◆ *MANTIS Entity Transformers*, P39-0013
- ◆ *MANTIS DL/I Programming, OS/390, VSE/ESA*, P39-5008
- ◆ *MANTIS SAP Facility, OS/390, VSE/ESA*, P39-7000
- ◆ *MANTIS WebSphere MQ Programming*, P39-1365
- ◆ *MANTIS Application Development Tutorial, OS/390, VSE/ESA*, P39-5026



Manuals marked with an asterisk (*) are listed twice because you use them for multiple tasks.

Educational material

AD/Advantage and MANTIS educational material is available from your regional Cincom education department.

1

Overview of MANTIS SQL support

MANTIS is an application development system for developing, testing, and executing applications interactively. MANTIS SQL Support is an extended version of MANTIS. It enables you to create MANTIS applications that access DB2 for OS/390 or DB2 for VSE and VM (formerly SQL/DS) databases by using SQL statements embedded in MANTIS programs. The presence of MANTIS SQL Support does not affect non-SQL MANTIS applications. MANTIS SQL Support programs can run side by side or with non-SQL MANTIS programs, with neither affecting the other.

Programming SQL in MANTIS SQL Support is similar to programming SQL in other programming languages. This manual refers to SQL in COBOL as representative of those languages. Because MANTIS is interpretive rather than compiled, some differences exist between MANTIS SQL Support and SQL in COBOL. These differences are noted in this manual and summarized in Appendix C.

Embedding SQL statements in MANTIS programs

You embed SQL statements in MANTIS programs as standard MANTIS comments. You must precede each SQL statement with an EXEC_SQL statement and follow it with an END statement, as shown below. Code the SQL statement text between these statements as MANTIS comments (each line must begin with a vertical bar (|), which is the MANTIS comment character.)

An example of an SQL SELECT statement in a MANTIS program is shown below. Note that MANTIS automatically sets the indentation level (number of preceding periods) for all the statements.

```
04590 ..TEXT EMPL_NAME(30)
04600 ..BIG EMPL_NAME_IV
04610 ..EXEC_SQL
04620 ... | SELECT EMPLNAME
04640 ... | INTO :EMPL_NAME:EMPL_NAME_IV
04650 ... | FROM EMPLOYEE.TABLE
04660 ... | WHERE EMPLNAME = 'SMITH'
04670 ..END
04680 ..DO CLEAN_UP
```

MANTIS variables can appear in SQL statements (they are required by some SQL statements). MANTIS variables coded in SQL statements are called **host variables**. They transfer data between MANTIS and the SQL database. Host variables can be followed by **indicator variables**, which indicate the presence of NULL or truncated data. A colon (:) must precede all host and indicator variables coded in SQL statements. Line 4640 above shows how host and indicator variables are used in SQL statements. See Chapter 2 for more information on host and indicator variables and embedding SQL statements in MANTIS programs.

Static and dynamic SQL statements

In this manual, the words “static” and “dynamic” are used in two different contexts: they both refer to the SQL statement type and the MANTIS SQL Support execution mode.

Use of static vs. dynamic SQL statements is explained below:

- ◆ **Static SQL statements.** Use static SQL statements when you know all the information required to execute the statement (SQL table name, column names, etc.) before executing the statement, and the information will not change. `SELECT`, `INSERT`, `UPDATE`, and `DELETE` are examples of static SQL statements. In SQL in COBOL, you must code these statements in an application program and precompile and compile them before executing them.
- ◆ **Dynamic SQL statements.** Use dynamic SQL statements to execute other SQL statements that have not been precompiled. Use dynamic SQL statements when you do not know all of the information needed to execute the SQL statement before executing the statement. An example is a query application that allows the user to input the SQL statement to be executed from an online terminal. `PREPARE`, `DESCRIBE`, `EXECUTE`, and `EXECUTE IMMEDIATE` are examples of dynamic SQL statements. In SQL in COBOL, you must precompile and compile dynamic SQL statements, but not the SQL statements they execute. Dynamic SQL statements provide flexibility, but require more computer resources and deliver less performance than static SQL statements.

The following sections explain static and dynamic execution modes.

MANTIS SQL support execution modes

MANTIS SQL Support executes programs in one of three modes:

- ◆ Dynamic execution mode (DB2 for VSE and VM—formerly SQL/DS—and DB2 for OS/390)
- ◆ Static execution mode (DB2 for OS/390 only)
- ◆ Extended dynamic execution mode (DB2 for VSE and VM only)



Do not confuse static and dynamic SQL statements with static and dynamic execution modes. Both static and dynamic statements can execute in static, dynamic, or extended dynamic execution modes.

Dynamic execution mode (DB2 for VSE and VM—formerly SQL/DS—and DB2 for OS/390)

In dynamic execution mode, MANTIS performs all the processing of each EXEC_SQL statement just as the statement is encountered. No precompiling occurs prior to execution. You can code SQL statements in a MANTIS program and immediately RUN the program. In this mode, all EXEC_SQL statements are executed using dynamic SQL statements (MANTIS translates static SQL statements into equivalent dynamic SQL statements before executing them). Dynamic execution mode is the default execution mode.

Static execution mode (DB2 for OS/390)

Static execution mode in MANTIS SQL Support is similar to SQL in COBOL. The SQLBIND process and SQL Generate program extract SQL statements from the MANTIS program into a BAL (Basic Assembler Language) source code module. This module is then precompiled and assembled as is a COBOL program. Instead of using dynamic SQL statements, the MANTIS program executes EXEC_SQL statements from the precompiled BAL module. This method of execution provides a considerable increase in performance and resource use over dynamic execution mode. Static execution mode is only available when using DB2 for OS/390. See “[Preparing MANTIS SQL support programs for static execution mode \(DB2 for OS/390\)](#)” on page 75 for more information on static execution mode.

Extended dynamic execution mode (DB2 for VSE and VM—formerly SQL/DS)

Extended dynamic execution mode is similar to static execution mode. The SQLBIND process extracts and compiles EXEC_SQL statements from MANTIS programs, providing the performance and resource advantages of static execution mode. However, unlike static execution mode, this process is done entirely online (static execution mode requires off-line processes such as precompiling and assembling). Extended dynamic execution mode uses extended dynamic SQL statements. These statements are not supported by DB2 for OS/390. Consequently, extended dynamic execution mode is available only to MANTIS programs accessing DB2 for VSE and VM. See [“MANTIS SQL support programs for extended dynamic execution mode \(DB2 for VSE and VM—formerly SQL/DS\)”](#) on page 113 for more information on extended dynamic execution mode.

The following chapter explains how to embed SQL statements in MANTIS programs for all of these execution modes.

2

Embedding SQL statements in MANTIS programs

Rules for embedding SQL statements in MANTIS programs

You embed SQL statements in MANTIS programs within an EXEC_SQL-END block. Begin each SQL statement with EXEC_SQL and terminate with END. You must begin each line of SQL text between EXEC_SQL and END with the MANTIS comment character, the vertical bar (|). All SQL text within the EXEC_SQL-END block must conform to the rules of SQL syntax (rather than MANTIS syntax), except where host language syntax is permitted.

When you embed SQL statements in a MANTIS program, the following rules apply:

- ◆ Only one SQL statement can be present within an EXEC_SQL-END block.

```
..EXEC_SQL  
... | OPEN C1  
... | FETCH C1 INTO ...  
... | CLOSE C1  
..END
```

Invalid: Three SQL statements in the EXEC_SQL-END block.

- ◆ Any text between EXEC_SQL and END must be part of an SQL statement and must be preceded by a vertical bar (|). Once MANTIS SQL Support encounters a vertical bar, the rest of the program line is considered part of a single SQL statement. Other MANTIS statements or comments are not permitted.

```
..EXEC_SQL
...| OPEN C1
...OPENED=TRUE
..END
```

Invalid: A statement other than a comment is between EXEC_SQL and END.

```
..EXEC_SQL
...| OPEN C1:OPENED=TRUE
..END
```

Invalid: A MANTIS statement is appended to a valid SQL statement.

```
..EXEC_SQL
...| OPEN C1:EMPLOYEE CURSOR
..END
```

Invalid: A comment is appended to a valid SQL statement.

- ◆ A colon within an EXEC_SQL-END block identifies a MANTIS host variable, not a new statement.

```
..BIG A
..EXEC_SQL
...| FETCH C1 INTO :A
..END
```

*C1 is an SQL entity;
A is a MANTIS host variable.*

- ◆ An SQL statement appended to an EXEC_SQL statement with a colon (the character that separates MANTIS statements) is part of the SQL statement; it is considered to be within the EXEC_SQL-END block.

```
..EXEC_SQL:| SELECT NAME
...| FROM EMPL_TABLE
...| WHERE ZIP = '12345'
..END
```

Valid

- ◆ In an SQL statement, multiple blanks at the beginning or end of an SQL statement are treated as a single blank.

```

..EXEC_SQL          is          ..EXEC_SQL
... | OPEN C1      equivalent   ... | OPEN C1
..END              to          ..END
    
```

All spaces between words on the same or different lines are compressed at every execution except those contained in a text literal.

```

..EXEC_SQL          is          ..EXEC_SQL
... |              equivalent   ... | SELECT COL1 FROM
... | SELECT       to          ... | TABLE WHERE
... | COL1        ..END       ... | COL1=' ABCDEF '
... | FROM TABLE
... | WHERE
... | COL1=' ABCDEF '
..END
    
```

Although multiple blanks can add to readability, they also incur additional processing overhead. You may want to avoid using them for this reason.

- ◆ An SQL statement in an EXEC_SQL-END block can be broken into multiple lines. MANTIS SQL Support reads the text on two consecutive comment lines in an EXEC_SQL-END block as if it were separated by a single blank (one SQL statement).

```

..EXEC_SQL          is          ..EXEC_SQL
... | OPEN         equivalent   ... | OPEN C1
... | C1          to          ..END
..END
    
```

- ◆ Do not code SQL text literals on more than one line.

```
..EXEC_SQL
...| SELECT COL1
...| INTO :HOST_VAR
...| FROM TABLE
...| WHERE COL1='ABC
...| DEF '
..END
```

Invalid: An SQL text literal appears on more than one line.

- ◆ A MANTIS statement on the same line as the END statement in an EXEC_SQL-END block is not executed. This is consistent with the rules for using END with MANTIS IF, WHILE, WHEN, and UNTIL statements. MANTIS comments are permitted.

```
..EXEC_SQL
...| OPEN C1
..END:OPENED=TRUE
```

OPENED=TRUE is disregarded.

```
..EXEC_SQL
...| OPEN C1
..END: C1 IDENTIFIES TAG FILE ENTRIES
```

A valid comment.

Coding host variables in SQL statements

Host variables are MANTIS data variables that are used to provide input or receive output from the SQL database. A colon (:) prefix identifies host variables in SQL statements. In the following example, EMPL is a host variable:

```
..SMALL EMPL
..EXEC_SQL
...| FETCH CURSOR1 INTO :EMPL
..END
```

You can explicitly declare host variables as BIG, SMALL, TEXT, or KANJI. Like other MANTIS variables, undeclared host variables are implicitly declared as BIG when they are first used. Any previously undefined MANTIS variable referenced in an SQL statement is automatically declared as a MANTIS BIG variable.

```
..BIG A                               is equivalent   ..EXEC_SQL
..EXEC_SQL                             to               ...| FETCH C1 INTO :A
...| FETCH C1 INTO :A                  ..END
..END
```

Host variables can either be input host variables or output host variables, depending on how they are used in the SQL statement. Input host variables contain data that the SQL database requires to perform the SQL statement; that is, they contain input for DB2. These are usually search condition values. EMPL2 is an input host variable in the example below. It contains the employee number of the specific employee to be selected.

Output host variables are variables that contain data requested from the SQL database; that is, they contain output from DB2. They are usually found in SELECT or FETCH statements. In the following example, EMPL1 is an output host variable since the SQL database will place the results of the SELECT statement there:

```
..TEXT EMPL1(30)
..SMALL EMPL2
..EXEC SQL
...| SELECT EMPLNAME
...| INTO :EMPL1                               Output host variable
...| FROM EMPL.TABLE
...| WHERE EMPLNO = :EMPL2                     Input host variable
..END
```

Normally, host variables are MANTIS variables declared as BIG, SMALL, TEXT, or KANJI. Certain MANTIS functions and complex data types (such as SCREEN and FILE) can also be used as host variables, depending on how they are used in the SQL statements. The following table shows the MANTIS entities that can be used as SQL host variables:

Declared data variables	Valid for input host variable	Valid for output host variable
BIG	yes	yes
SMALL	yes	yes
TEXT	yes ¹	yes ¹
KANJI	yes ¹	Yes ¹
Immediate numeric functions:		
DATAFREE	yes	no
DOLEVEL	yes	no
E	yes	no
FALSE	yes	no
PI	yes	no
PROGFREE	yes	no
TRUE	yes	no
USERWORDS	yes	no
ZERO	yes	no
Immediate text functions:		
DATE KEY	yes	no
PASSWORD	yes	no
PRINTER	yes	no
TERMINAL	yes	no
TERMSIZE	yes	no
TIME	yes	no
USER	yes	no

¹ Input host variables can include both array and substring subscripts. Output host variables may include array subscripts but cannot include substring subscripts.

Declared data variables	Valid for input host variable	Valid for output host variable
Complex data types:		
ACCESS	yes ²	no
FILE	yes ²	no
INTERFACE	yes ²	no
SCREEN	yes ²	no
TOTAL	yes ²	no
VIEW	yes ²	no
ENTRY	no	no
PROGRAM	no	no

² These are treated as TEXT variables.

A host variable can be located in a MANTIS array. You can use arithmetic expressions and MANTIS functions to specify subscripts of host variables. MANTIS syntax rules apply to subscripting, even though the subscript is coded in an SQL statement. In the example below, all text following the colon must conform to MANTIS syntax rules. For example:

```
..SMALL N,T
..BIG EMPL(20,40)
..EXEC_SQL
...| FETCH C1 INTO :EMPL(1+N,INT(T))
..END
```



Prefix only the host variable with a colon, not the other MANTIS variables referred to in subscript expressions. In the example above, the variables N and T are not prefixed with a colon, but are MANTIS variables used to calculate the subscript for the EMPL array.

Accessing host variables directly from within arrays is an extension to SQL made for MANTIS SQL Support. It may not be available in SQL in COBOL.

You can use host variables in SQL expressions. A colon, as shown in the following example, must precede each host variable:

```
EXEC_SQL
..| INSERT INTO OWNER.TAB (COL-A)
..| VALUES (:SALARY * 1.1)
..END
```

Coding indicator variables in SQL statements

Indicator variables are host variables that contain information about the data being sent to or received from the SQL database. Indicator variables are used to indicate whether the data transferred between MANTIS and the SQL database was transferred successfully, is NULL, or was truncated.

Indicator Variable Value	Description
= 0	Successful transfer.
< 0	Data was NULL on database.
> 0	Data truncated. Value indicates untruncated length.

Indicator variables are optional, but if used, they must be prefixed with a colon and immediately follow the corresponding host variable (or subscript expression). In the following example, EMPLIV and NAMEIV are indicator variables:

```
..EXEC_SQL: | SELECT EEMPLNO, EEMPLNA
... | INTO :EMPL(15,3):EMPLIV, :NAME:NAMEIV
... | FROM EMPLOYEES WHERE DEPT = 17
..END
```

Like host variables, indicator variables can be defined explicitly or implicitly. Only SMALL and BIG variables can be used as indicator variables. The default in implicit declaration is a MANTIS BIG variable. You can also use an indicator variable to insert a NULL value into the SQL database. Storing a negative value in an indicator variable causes the SQL database to make the value of the associated table column NULL, regardless of the contents of the host variable. In the following example, a NULL value is inserted for column COLA (regardless of the value of VAR) when the INSERT statement is executed:

```
..VARIV=(-1)
..EXEC_SQL
... | INSERT INTO OWNER.TAB (COLA,COLB)
... | VALUES (:VAR:VARIV, :XV:XYIV)
..END
```

Indicator variables are required to set an SQL table column to NULL, as in the example above.

When accessing the SQL database, you may need to check the indicator variable before using the data returned by the database. If a table column is NULL, the value of the associated host variable is not defined. The contents of the MANTIS host variable may be unchanged or may be cleared, depending on options selected when MANTIS was installed. Check with your system administrator for more information.

Converting data between MANTIS SQL support and the SQL database

MANTIS SQL Support data is always maintained in MANTIS data types (BIG, SMALL, TEXT, KANJI). If data conversion is required, the SQL database performs it, unless your program requests MANTIS SQL Support to convert the data type (see [“Specifying SQL data types in host variables”](#) on page 34). The following table lists permissible data type conversions. Any combination of MANTIS and SQL data types not listed in this table may result in run-time errors. Note that loss of precision, numeric overflow, and data truncation are possible during data conversion.

The following table shows valid data type conversions:

SQL data type	MANTIS data type	Considerations
DECIMAL	BIG, SMALL	Loss of precision may occur when converting from SQL to MANTIS. Overflow may occur when converting from MANTIS to SQL.
INTEGER	BIG, SMALL	Loss of precision may occur when converting from SQL to MANTIS. Overflow may occur when converting from MANTIS to SQL.
SMALLINT	BIG, SMALL	Overflow may occur when converting from MANTIS to SQL.
FLOAT	BIG, SMALL	Overflow and/or loss of precision may occur when converting from SQL to MANTIS SMALL variables.
CHAR VARCHAR LONG VARCHAR	TEXT	Truncation may occur in either direction. (Check the indicator variable.)
GRAPHIC VARGRAPHIC LONG VARGRAPHIC	KANJI	Truncation may occur in either direction. (Check the indicator variable.)
DATE	TEXT	Truncation may occur if TEXT size is less than 10.
TIME	TEXT	Truncation may occur if TEXT size is less than 8.
TIMESTAMP	TEXT	Truncation may occur if TEXT size is less than 26.



The SQL data types LONG VARCHAR and LONG VARGRAPHIC are not fully supported in MANTIS SQL Support. They are treated in the same way as the SQL types VARCHAR and VARGRAPHIC: they are supported as TEXT or KANJI variables, having a maximum of 254 characters (TEXT) or 127 characters (KANJI).

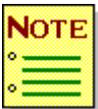
Specifying SQL data types in host variables

MANTIS SQL Support always maintains data in MANTIS data types (BIG, SMALL, TEXT, KANJI). Data transferred to the SQL database is presented in the SQL equivalent to these data types (FLOAT, VARCHAR, VARGRAPHIC). The SQL database then performs any data conversions required (see “[Converting data between MANTIS SQL support and the SQL database](#)” on page 32).

In some circumstances, conversion by the SQL database is not desirable. DB2 for OS/390 and DB2 for VSE and VM may not use indexes to search SQL tables if the data type of the search condition value is different from that of the table column. For example:

```
..BIG EMPLOYEE_NUM
..EMPLOYEE_NUM=100
..EXEC_SQL
...| DECLARE C1 CURSOR FOR
...| SELECT FROM EMPLNO,EMPLNA
...| FROM EMPLOYEE.TABLE
...| WHERE EMPLNO> :EMPLOYEE_NUM
..END
```

If the SQL data type of the EMPLNO table column is INTEGER, an index defined for this column may not be used to perform the SELECT because MANTIS presents the value of EMPLOYEE_NUM to the SQL database in the SQL data type equivalent of BIG, which is FLOAT. Because this data type is not the same as the EMPLNO column (INTEGER), DB2 may not use an index for the EMPLNO table column.



This condition only occurs in static or extended dynamic execution modes. It does not occur in dynamic execution mode. This condition is caused by the DB2 database, not by MANTIS SQL Support.

MANTIS SQL Support allows your program to specify the SQL data type for each host variable in an SQL statement. When you specify the SQL data type, MANTIS SQL Support converts the host variable data to the requested SQL data type before calling the SQL database.

You specify the SQL data type for a host variable by adding the SQL data type specification to the host variable name in the SQL statement. You must delimit it with MANTIS comment characters (vertical bars) and it must precede any subscript or indicator variable definitions. You can specify conversion only for numeric data types. The conversion must conform to the same rules for data type declaration as those in the SQL CREATE TABLE statement. The table at the end of this section shows the valid SQL data type specifications.

To enable the SQL database to use any available index on the SELECT statement in the previous example, code the following:

```

..EXEC_SQL
...| DECLARE C1 CURSOR FOR
...|   SELECT EMPLNO,EMPLNA
...|     FROM EMPLOYEE.TABLE
...|     WHERE EMPLNO>:EMPLOYEE_NUM|INTEGER|
..END

```

The |INTEGER| added to the EMPLOYEE_NUM host variable name causes MANTIS SQL Support to convert the value of EMPLOYEE_NUM from FLOAT to INTEGER before executing the SELECT statement. When the statement is executed, the same data types are in both the search condition value (EMPLOYEE_NUM) and the table column, so DB2 can use any available index.

You can specify an SQL data type for any host variable in any SQL statement. However, it is only significant for input host variables (it is ignored for output host variables) and only needed in search conditions (WHERE clauses).

When you specify an SQL data type, MANTIS SQL Support converts the data in the host variable to the specified SQL data type and presents it to the SQL database in this form. This process may introduce slight variation in the data because the numeric conversion routines used by MANTIS SQL Support are not the same ones used by the SQL database. Also note that SQL data type specifications are only needed for programs that execute in static or extended dynamic execution mode (see [“Preparing MANTIS SQL support programs for static execution mode \(DB2 for OS/390\)”](#) on page 75 and [“MANTIS SQL support programs for extended dynamic execution mode \(DB2 for VSE and VM—formerly SQL/DS\)”](#) on page 113). Programs executing in dynamic execution mode do not require the SQL data type specification in order to use an available index (these restrictions are imposed by DB2, not by MANTIS SQL Support).

In the table below, “pp” and “ss” refer to one or two numeric digits. The “pp” characters are numeric precision and must be included where they appear. The “ss” characters are numeric scale digits and are optional. With the exception of coding one or two numeric digits (where applicable), the syntax must be coded as shown, with no intervening blanks.

The SQL data type specification must be delimited by the vertical bar (|) character and must immediately follow the host variable name, before any subscript and/or indicator variable expressions. The SQL data type specifications are shown below with delimiters included.

SQL data type	Specification
DECIMAL(pp,ss) or DEC(pp,ss)	0 <= pp < 15; 0 <= ss <= pp
INTEGER or INT	No “pp” or “ss” permitted
SMALLINT	No “pp” or “ss” permitted
FLOAT(pp) or FLOAT	0 <= pp < 53; pp < 22 for REAL pp >= 22 for DOUBLE PRECISION
DOUBLE PRECISION	Double precision floating point (same as MANTIS BIG)
REAL	Single precision floating point (same as MANTIS SMALL)

3

Programming considerations

MANTIS SQL Support allows you to execute SQL statements from a MANTIS program. SQL statements are coded in a MANTIS program as standard MANTIS comments, enclosed in an EXEC_SQL-END block. As MANTIS encounters each SQL statement, it prepares it for execution and then executes it, in effect performing the same steps (preprocess, compile, link, and execute) that are performed for COBOL programs that contain embedded SQL statements. However, unlike COBOL programs, MANTIS programs can be modified (including the SQL statements) and then immediately re-executed.

Before you begin writing MANTIS SQL Support programs, review the programming considerations discussed in this chapter:

- ◆ In dynamic execution mode, a limited number of SQL statements can be active concurrently. This number varies, depending on the SQL database in use and the limit set by your administrator. (“[SQL statement limits](#)” on page 39)
- ◆ The scope of an SQL cursor or statement is local to a program. Because both are SQL entities and not MANTIS entities, you cannot pass them as parameters or use them in non-SQL MANTIS statements. (“[Scope of SQL cursors and statements](#)” on page 40)
- ◆ The WHENEVER statement in MANTIS SQL Support differs slightly from the WHENEVER statement in other SQL in COBOL. (“[SQL WHENEVER statement](#)” on page 40)
- ◆ Elements in the SQLCA (SQL Communications Area) are accessed through a MANTIS function called SQLCA, rather than as elements of an SQLCA data structure as in SQL in COBOL. (“[SQLCA in MANTIS SQL support](#)” on page 45)
- ◆ The effects of the SQL COMMIT WORK and ROLLBACK WORK statements are identical to the COMMIT and RESET statements in MANTIS. Executing an SQL COMMIT or ROLLBACK causes a MANTIS COMMIT or RESET to be executed and vice versa. (“[SQL COMMIT WORK and ROLLBACK WORK statements](#)” on page 50)
- ◆ Any SQL statement or function (EXEC_SQL, SQLCA, SQLDA) terminates MANTIS HPO (High Performance Option) binding. (“[Binding with the High-Performance Option \(HPO\)](#)” on page 51)
- ◆ Running a program from a line number can have unpredictable results. (“[Running a program from a line number](#)” on page 51)
- ◆ Error messages can come from three different sources: MANTIS SQL Support, the MANTIS nucleus, and the SQL database. (“[Error messages](#)” on page 52)
- ◆ MANTIS SQL Support does not limit the number of host variables in an SQL statement. However, this number may be limited by DB2. (“[Maximum number of host variables](#)” on page 54)
- ◆ Certain SQL keywords cannot be used as MANTIS host variable names. (“[SQL keywords](#)” on page 155)
- ◆ Truncation of TEXT data can occur when parameters of MANTIS ENTRY statements are used as host variables in SQL statements. (“[Using ENTRY statement parameters as host variables](#)” on page 54)

SQL statement limits

In dynamic execution mode only, a fixed number of SQL statements can be used concurrently. This limit is set for your site when MANTIS SQL Support is installed. The permitted range for DB2 for OS/390 is 0-512 and for DB2 for VSE and VM is 1-510. The default is 10. Your system administrator can change this number.



This SQL statement limit applies only when executing in dynamic execution mode. Programs executing in static or extended dynamic mode are not subject to this limit.

This limit represents the maximum number of SQL statements that can be concurrently active in your MANTIS program. It is not the maximum number of SQL statements your program can contain. Your program can contain as many SQL statements as you wish, but only this maximum can be active at one time.

In general, each SQL statement in a program counts as one against this maximum. However, all statements using the same cursor count only as a single statement against the maximum. Thus, the following code sequence uses one statement, not four, as applied against the maximum number of SQL statements:

```
DECLARE C1 CURSOR
OPEN C1
FETCH C1
CLOSE C1
```

This maximum exists for each MANTIS program executed, and applies to all DO levels for that program. Once assigned, an SQL statement is active until the Logical Unit of Work (LUW) terminates.

If you exceed the maximum permitted number of statements, you will receive this message:

```
NUCQMAE:TOO MANY STATEMENTS CONCURRENTLY IN USE
```

Scope of SQL cursors and statements

The scope of an SQL cursor or statement is local. An SQL cursor or statement cannot be referenced outside the MANTIS program where it was defined. Because statements and cursors are SQL entities, not MANTIS entities, they cannot be passed as parameters or used in non-SQL MANTIS statements.

Because SQL cursors and statements are local, the name of an SQL cursor or statement can be used in other programs without conflict. However, the same SQL cursor or statement defined in different MANTIS programs each count as one against the maximum number of SQL statements permitted (see “SQL statement limits” on page 39).

SQL WHENEVER statement

The WHENEVER statement in MANTIS SQL Support differs from the WHENEVER statement in SQL in COBOL in four ways. In MANTIS SQL Support:

- ◆ The WHENEVER statement is interpretive, not compiled.
- ◆ GOTO is replaced by DO.
- ◆ STOP is replaced by FAULT.
- ◆ The default action for SQLERROR is FAULT, not CONTINUE.

The syntax of the MANTIS SQL Support WHENEVER statement is shown below. Note that any action (DO, FAULT, or CONTINUE) can be selected for any condition (SQLERROR, SQLWARNING, NOT FOUND).



IF you do not code the WHENEVER statement, the conditions will default to the actions specified in the considerations below. If you do code the WHENEVER statement, you must code both condition and action.

WHENEVER	{	SQLERROR	}	{	DO	}
		SQLWARNING			FAULT	
		NOT FOUND			CONTINUE	

SQLERROR

Description *Optional.* Specifies that the SQL database has detected an error. The SQL statement execution was not successful and the SQLCA SQLCODE contains a negative value.

Consideration If you do not specify a WHENEVER statement, the SQLERROR default action is FAULT.

SQLWARNING

Description *Optional.* Specifies that the SQL database detected a condition that may require program intervention. The SQL statement execution was successful. The SQLCA SQLCODE may contain a positive value other than 100 and/or one or more of the SQLCA SQLWARN flags may contain nonblank characters.

Consideration If you do not specify a WHENEVER statement, the SQLWARNING default action is CONTINUE.

NOT FOUND

Description *Optional.* Specifies that the SQL database cannot find a row to satisfy your SQL statement, or there are no more rows to be retrieved. The SQLCA SQLCODE contains 100.

Consideration If you do not specify a WHENEVER statement, the NOT FOUND default action is CONTINUE.

DO

Description *Optional.* Specifies a standard MANTIS DO (internal or external) and corresponds to the WHENEVER-GOTO SQL statement in SQL in COBOL. WHENEVER-DO transfers control to the specified internal routine or external program whenever the named condition is encountered.

Considerations

- ◆ WHENEVER-DO can transfer control to an internal routine or external program, which in turn can contain any MANTIS logic, including CHAIN, EXIT, or STOP statements. The current values of any DO arguments are passed to the named subroutine or external program. The subroutine or external program EXIT returns control to the next statement following the EXEC_SQL that caused the DO to occur.
- ◆ The WHENEVER-DO action resembles the existing functionality of the TRAP statement in MANTIS. If the DO portion of a WHENEVER-DO contains an error, MANTIS returns a MANTIS error message associated with the DO statement, not an SQL WHENEVER-type error. MANTIS displays the line in error in the subroutine or external program. The WHENEVER statement may be outside of the current execution path. Remember that DO is executed as a result of an SQL statement detecting the condition with which the DO action is associated.

FAULT

Description *Optional.* Terminates execution of the program and displays the error message returned by the SQL database in the form of a MANTIS fault message. FAULT corresponds to the WHENEVER-STOP SQL statement in SQL in COBOL.

CONTINUE

Description *Optional.* Permits program execution to continue without interruption when the named condition occurs. Program execution continues with the statement following the EXEC_SQL statement in the MANTIS program. Your program should then check the SQLCA SQLCODE for the results of each SQL statement execution.

The following table shows the default action for each condition when the WHENEVER statement is not coded:

Condition	Default action
SQLERROR	FAULT
SQLWARNING	CONTINUE
NOT FOUND	CONTINUE

Example

```
0230 EXEC_SQL: WHENEVER SQLERROR DO DO_ROUTINE(PARM1, PARM2, PARM3)
00240 END
00250 EXEC_SQL: WHENEVER SQLWARNING FAULT
00260 END
00270 EXEC_SQL: WHENEVER NOT FOUND CONTINUE
00280 END
```

Using SQL WHENEVER as a declarative statement

In SQL in COBOL, the SQL WHENEVER statement is a declarative statement. It is processed when the program is precompiled, not when it is executed. Consequently, in SQL in COBOL the current SQL WHENEVER setting is determined by its sequential position in the program.

In MANTIS SQL Support, the WHENEVER statement is an executable statement. The last executed WHENEVER statement is in effect regardless of its sequential position in the program. This difference is important when a WHENEVER statement is used with conditional statements. The following figure illustrates the different effects of a declared versus executed WHENEVER statement. C denotes a condition and 1 and 2 denote actions. The same considerations apply to UNTIL, WHEN, and IF structures in MANTIS.

SQL in COBOL pseudocode:	Setting in effect	MANTIS SQL support pseudocode:	Setting in effect
20 WHENEVER C1	C1	20 WHENEVER C1	C1
40 WHILE	C1	40 WHILE condition	C1 FIRST, THEN C2
50 WHENEVER C2	C2	50 WHENEVER C2	C1 or
70 ENDWHILE	C2	70 ENDWHILE	C1 or
80 EXEC_SQL	C2	80 EXEC_SQL	C1 or C2, depending on whether initial condition is FALSE or TRUE

Since the setting is established before run time, it remains unchanged regardless of whether lines 50-70 are executed.

The first time statement 40 is executed, the setting is C1; thereafter it is C2.

*However, if the WHILE condition is not true the first time line 40 is executed, C1 remains in effect through line 80 because line 50 was not executed.

Scope of the WHENEVER statement

The scope (range) of the WHENEVER statement is every EXEC_SQL statement in the current MANTIS DO-level until a new WHENEVER statement is executed. To change from the default WHENEVER settings, explicitly execute the WHENEVER statement in every external subprogram.

SQLCA in MANTIS SQL support

In SQL in COBOL, the SQLCA (SQL Communications Area) is a data structure. SQL in COBOL accesses elements in the SQLCA as items of data. In MANTIS SQL Support, these elements are accessed through the SQLCA statement and function.

The SQLCA statement and function move data to and from elements of the SQLCA structure.

SQLCA statement syntax

The SQLCA statement stores data from the MANTIS program into the SQLCA.

SQLCA(sqlca_element_name)=expression

sqlca_element_name

- | | |
|----------------------|---|
| Description | <i>Required.</i> Specifies the element of the SQLCA that is to receive data. |
| Format | Must be a text literal or expression that evaluates to one of the SQLCA element names in the following tables. |
| Consideration | Element names must be selected from the list for the SQL database in use (see the table under “ SQLCA elements ” on page 48). |

expression

- | | |
|----------------------|---|
| Description | <i>Required.</i> Specified the data to be transferred into the SQLCA. |
| Format | Must be consistent with the data type of the SQLCA element, text or numeric. |
| Consideration | Certain SQLCA elements are read-only and cannot have data stored into them by the MANTIS program. |

General considerations

- ◆ Some SQLCA elements are not present in the DB2 SQLCA. These are extensions to the SQLCA unique to MANTIS. They are: DBTYPE, MSGTEXT, and SQLISL.
- ◆ Although data can be stored in some SQLCA elements, doing so does not pass any information to the SQL database. The SQLCA is returned to the MANTIS program after each SQL statement is executed (EXEC_SQL-END). Any data stored in the SQLCA will be overwritten when the next SQL statement is executed. You can save and then restore the SQLCODE value and use it to retrieve the MSGTEXT value.

Example

This example shows how a MANTIS program can retrieve SQL error message text for an SQLCA error message. The SQLCA statement is used to store the SQLCODE value in the SQLCA, and the MSGTEXT function is used to retrieve the error text.

```
00150 TEXT SQL_ERROR_TEXT(254)
00160 SQLCA("SQLCODE")=(-504)
00170 SQL_ERROR_TEXT=SQLCA("MSGTEXT")
00180 END
```

SQLCA function syntax

The SQLCA function, shown below, transfers data from the SQLCA into the MANTIS program.

SQLCA(sqlca_element_name)

sqlca_element_name

Description *Required.* Specifies the element of the SQLCA that is to be transferred.

Format Must be a text literal or expression that evaluates to one of the SQLCA element names in the table under “SQLCA elements” on page 48.

General considerations

- ◆ Some SQLCA elements are not present in the SQL SQLCA. These are extensions to the SQLCA unique to MANTIS SQL support. They are DBTYPE, MSGTEXT, and SQLISL.
- ◆ If you move an SQLCA TEXT element to a MANTIS variable of shorter length (for example, an 8-character SQLCA element to a 6-character MANTIS variable), the right-most characters are truncated.

Example This example shows how data is retrieved from the SQLCA by the SQLCA function. Line 160 checks the SQLCA SQLCODE to determine if all table rows have been fetched.

```
00130 EXEC_SQL
00140 .| FETCH C1 INTO :EMPL_NAME, :EMPL_NAME
00150 END
00160 IF SQLCA("SQLCODE")=100
00170 .DO END_OF_DATA
00180 END
```

SQLCA elements

The following table lists SQLCA elements, the compatible MANTIS variable type, and usage notes:

Element name	MANTIS compatible data type	Contents / considerations	Updateable?
SQLCAID	TEXT(8)	Eyecatcher. Set by SQL.	No
SQLCABC	BIG	Length of SQLCA. Set by SQL.	No
SQLCODE	BIG	Code indicating results of SQL statement execution. For possible returned values, refer to your DB2 manual.	Yes
SQLERRM ¹	TEXT(70)	Tokens for insertion into SQL error message text. The vertical bar () replaces hexadecimal "FF" as the separator character.	Yes
SQLERRP	TEXT(8)	SQL diagnostic data.	Yes
SQLERRDn	BIG	SQL diagnostic data. n ranges between 1-6.	Yes
SQLWARNn	TEXT(1)	SQL warning flags. n ranges between 0–A.	Yes
SQLSTATE ²	TEXT(5)	Indicates the results of SQL statement execution.	Yes
SQLEXT	TEXT(8) DB2 TEXT(5) DB2 for VSE and VM	Reserved for SQL.	Yes

Element name	MANTIS compatible data type	Contents / considerations	Updateable?
DBTYPE ³	TEXT(6)	<p><i>MANTIS SQL extension.</i> DBTYPE returns the SQL database currently in use:</p> <ul style="list-style-type: none"> ◆ “DB2” for DB2 for OS/390 ◆ “SQL/DS” for SQL/DS or DB2 for VSE and VM 	No
MSGTEXT ³	TEXT(254)	<p><i>MANTIS SQL extension.</i> MSGTEXT returns the SQL error message text associated with the current SQLCA SQLCODE.</p>	No
SQLISL ³	TEXT(1)	<p><i>MANTIS SQL extension.</i> DB2 for VSE and VM variable. SQLISL allows the MANTIS program to set the SQL/DS or DB2 for VSE and VM ISOLATION parameter. Not supported by DB2 for OS/390. For more information about SQLISL, refer to the IBM DB2 for VSE and VM documentation.</p>	Yes

- ¹ If SQLERRM contains multiple character strings and you then show the contents of SQLERRM, the character strings will be separated by the vertical bar (|). You can use the POINT function and substring capabilities of MANTIS to split the message for display purposes. MANTIS changes any vertical bar character (|) found in the string to a broken vertical bar character (|).
- ² SQLSTATE is supported only for DB2 version 2.3 and higher, or SQL/DS version 3.4 and higher.
- ³ This element is a MANTIS extension to the SQLCA. It is not present in the DB2 SQLCA.

SQL COMMIT WORK and ROLLBACK WORK statements

In MANTIS SQL Support, the SQL COMMIT WORK and ROLLBACK WORK statements have exactly the same effect on the SQL database as the MANTIS COMMIT and RESET statements. An SQL COMMIT WORK or ROLLBACK WORK implies a MANTIS COMMIT and RESET, and vice versa. Executing an SQL COMMIT WORK or ROLLBACK WORK statement affects both the SQL database and all resources known to MANTIS.

In CICS, MANTIS automatically performs a COMMIT at terminal output. A CICS SYNCHPOINT is executed at each of the following:

- ◆ Terminal output, unless COMMIT OFF is in effect
- ◆ COMMIT
- ◆ RESET
- ◆ PERFORM, when you perform a routine that returns with a new transaction ID

Binding with the High-Performance Option (HPO)

You can HPO bind MANTIS SQL Support programs using the MANTIS BIND command. The binding process stops when EXEC_SQL, SQLCA, or SQLDA is encountered because they are not bindable statements. The SQLCA function is discussed in “SQLCA in MANTIS SQL support” on page 45. Binding (part of the MANTIS High-Performance Option) is discussed in *MANTIS Program Design and Editing, OS/390, VSE/ESA*, P39-5013.

Running a program from a line number

In MANTIS SQL Support you can run programs from a line number. However, this action can produce unpredictable results when the program contains SQL statements. Some SQL statements (such as FETCH) require that other SQL statements (such as DECLARE and OPEN) be previously executed. Running a program from a line number can cause errors because SQL statements have not been executed in the proper sequence. Also, SQL entities (cursors) can be affected by the MANTIS program display when running in MANTIS programming mode (the screen display causes a terminal I/O COMMIT, which causes an SQL COMMIT WORK, which can cause all SQL cursors to be closed).

Error messages

When using MANTIS SQL, you can receive messages from three sources:

- ◆ MANTIS nucleus
- ◆ MANTIS SQL Support
- ◆ SQL database

Messages from the MANTIS nucleus and SQL support are documented in *MANTIS Messages and Codes, OS/390, VSE/ESA*, P39-5004.

Messages from the DB2 database are documented in the appropriate DB2 manuals.

MANTIS SQL Support messages have a seven-character code like other MANTIS error messages. All MANTIS SQL error codes contain the letter “Q”, as shown in the example below:

```
NUCQFKE: SPECIFIED SQLDA HOST VARIABLE ELEMENT IS UNINITIALIZED.
```

The way MANTIS SQL Support messages are displayed depends on whether the error was detected in the MANTIS Full-Screen Editor, the MANTIS Line Editor, or a MANTIS program:

- ◆ **In the MANTIS Full-Screen Editor.** If the message is too long to be displayed in full on the Message Line at the top of the screen, the message is displayed in a separate window at the bottom of the screen.
- ◆ **In the MANTIS Line Editor.** The MANTIS statement where the error was encountered is displayed at the bottom of the edit screen, followed by the error code and message. If the message is too long to fit on one line, the remaining text is not displayed.
- ◆ **In a MANTIS program.** The message appears at the bottom of the screen. It wraps to succeeding lines if necessary. The number of the statement causing the error and the MANTIS program name appear beneath the error message.

Messages from the SQL database contain the 3-character code QDB. A message from the database contains the SQLCA SQLCODE value and its associated text message. The format is:

```
NUCQDBE:±nnnnn:message-text
```

where ±nnnnn is the SQLCA SQLCODE value and message-text is the message returned from the database. For example:

```
NUCQDBE:-105:INVALID STRING CONSTANTS
```

is returned from DB2 if an SQLCODE of -105 is returned to the SQLCA and the WHENEVER SQLERROR condition was set to FAULT.

To display error message text for SQL database errors, MANTIS SQL Support must have access to certain SQL resources. For DB2, the SQL error message text modules (DSNTIAR, DSNTIAM) must be included when MANTIS is installed. For DB2 for VSE and VM, MANTIS must have authority to access the SQLDBA.SYSTEXT1 and SQLDBA.SYSTEXT2 tables. If MANTIS SQL Support cannot access the SQL error message text, the following message is displayed:

```
NUCQDBE:±nnnnn: AN SQL ERROR OR WARNING HAS OCCURRED  
SQLERRM: xxx|xxx...xxx
```

The second line contains the contents of the SQLCA SQLERRM element. This line will not appear for messages for which no SQLCA SQLERRM was returned.

Maximum number of host variables

The number of host variables that can be defined in a MANTIS program is limited only by DB2 for OS/390 or DB2 for VSE and VM. MANTIS SQL Support does not limit the number of host variables that can be defined in a MANTIS program (apart from the limit of 2048 symbolic names in a single MANTIS program).

Using ENTRY statement parameters as host variables

If ENTRY statement parameters are used as host variables in SQL statements in a MANTIS program, be certain that the length of any TEXT parameters passed do not change from one execution to the next. If this is not done, the data for TEXT parameters can be truncated. MANTIS SQL Support will use and maintain the length received on the first execution. If a TEXT parameter of a different length is used on a subsequent execution, the new length will be ignored. If the new length is larger than the existing length, MANTIS will continue to return the existing length, in effect truncating the data. The following example shows ENTRY parameters used as host variables:

```

10120 ENTRY GET_NAME(EMPL_NAME,EMPL_NUMBER)
10130 .EXEC_SQL
10140 ..| FETCH C1 INTO :EMPL_NAME, :EMPL_NUMBER
10150 .END
10160 EXIT

```

This problem can be eliminated by moving variables passed as ENTRY statement parameters to variables explicitly defined within the MANTIS program or subroutine. Then use the explicitly defined variables or host variables in the SQL statements, as shown in the following example:

```

10110 ENTRY GET_NAME(EMPL_NAME,EMPL_NUMBER)
10120 .TEXT NAME(30),NUMBER(12):| define as maximal size
10130 .EXEC_SQL
10140 ..| FETCH C1 INTO :NAME, :NUMBER
10150 .END
10160 .EMPL_NAME=NAME
10170 .EMPL_NUMBER=NUMBER
10170 EXIT

```

4

Dynamic SQL statements

Using dynamic SQL statements in MANTIS SQL Support is somewhat different than using dynamic SQL statements in SQL in COBOL. If you are unfamiliar with dynamic SQL statements, you may want to review the appropriate SQL database manuals for more information before reading this chapter.

Static and dynamic are the two basic types of SQL statements. You can use static SQL statements when you know all the information needed to execute the SQL statement before executing it. SQL SELECT, FETCH, UPDATE, INSERT, and DELETE are examples of static SQL statements. Use dynamic SQL statements when you do not know all the necessary information about an SQL statement before executing it. For example, if a query application allows a user to enter an SQL statement from an online terminal, the query program requires dynamic SQL statements. The DB2 SPUFI and DB2 for VSE and VM ISQL utility programs are examples of such applications.

Dynamic SQL statements allow you to execute other SQL statements under control of the application program. PREPARE, DESCRIBE, EXECUTE, and EXECUTE IMMEDIATE are examples of dynamic SQL statements. Alternative forms of the DECLARE, OPEN, and FETCH statements can also be used with dynamic SQL statements.

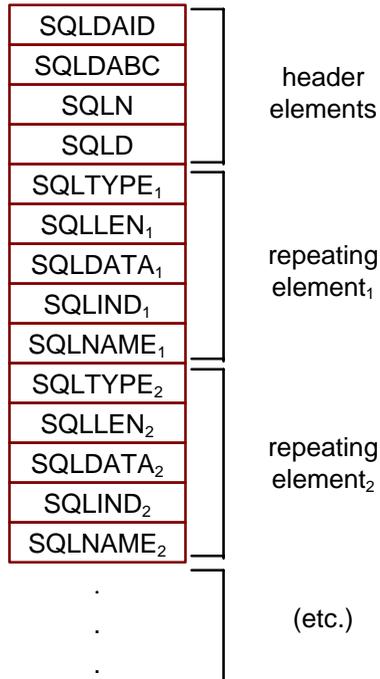
You can use both static and dynamic SQL statements in the same MANTIS program and in MANTIS programs executing in dynamic, static, or extended dynamic execution modes.

You cannot use host variables in SQL statements that are executed by dynamic SQL statements. A parameter marker (usually the question mark character (?)) must replace all host variables in the SQL statement text. Data is transferred between MANTIS and the SQL database using an SQLDA (SQL Descriptor Area). When using dynamic SQL statements, you must procedurally place host variable data into or retrieve data returned by the SQL database from an SQLDA.

To execute SQL statements using dynamic SQL, you must prepare the SQL statements with an SQL PREPARE statement and then execute them with an SQL EXECUTE statement. If data is retrieved, inserted, or updated in the SQL database, your program must manipulate an SQLDA between SQL statement preparation and execution. This manipulation can include allocating and expanding an SQLDA, retrieving data type and length information from the SQL database using the DESCRIBE statement, and transferring data between MANTIS variables and an SQLDA. Appendix A shows examples of MANTIS programs using static SQL statements and equivalent programs using dynamic SQL statements.

SQLDA statement and function

An SQLDA (SQL Descriptor Area) is a data structure that is used to transfer data between your program and the SQL database when dynamic SQL statements are used. The following figure shows the structure of an SQLDA:



An SQLDA is composed of two types of elements: header elements (which occur once per SQLDA) and repeating elements (which can occur multiple times in an SQLDA). Repeating elements repeat as a group (each element occurs only once in a group). This repeating group is called an SQLVAR. Each element in the SQLDA has a specific name and contains a specific item of data. These items are explained in the SQL documentation for the SQL database in use.

When you execute SQL commands in a COBOL program, you must explicitly declare each SQLDA element as a data area in your program and then access SQLDA elements through programming statements. In MANTIS SQL Support, you access the SQLDA by using the SQLDA statement and function. In MANTIS SQL Support, when you declare an SQLDA, an SQLDA with all the elements shown in the preceding illustration is built for you. The SQLDA contains a default number of repeating elements (SQLVARS). Your system administrator sets the default, but you can modify this number in your program.

SQLDA is both a statement and a function. The SQLDA statement (write) stores data from the MANTIS program into the SQL Descriptor Area (SQLDA). The SQLDA function (read) transfers data from the SQLDA into the MANTIS program.

The SQLDA statements are shown below followed by the SQLDA function. The SQLDA statement is used to allocate or deallocate an SQLDA, and to transfer data from a MANTIS program into an SQLDA.

Allocate an SQLDA

SQLDA(sqlda_name)=NEW

sqlda_name

- Description** *Required.* Specifies the name of the SQLDA to be allocated.
- Format** Must be a text literal or expression of 1–18 characters.
- Consideration** Naming conventions for SQL entities must be followed. Refer to the appropriate SQL language manual for the SQL database in use.
- Example** The following example shows how to allocate an SQLDA. Line 150 allocates an SQLDA named the “SQLDA1”.

```

00100 TEXT SQL_TEXT(254)
00110 TEXT EMPL_NAME(30),EMPL_STREET(30),EMPL_STATE(2)
00120 BIG EMPL_ZIP_CODE
00130 SQL_TEXT="SELECT NAME, STREET, STATE, ZIPCODE"
00140 SQL_TEXT=SQL_TEXT+" FROM EMPLOYEE.TABLE"
00150 SQLDA("SQLDA1")=NEW
00160 IF SQLDA("SQLDA1","SQLN")<4
00170 .SQLDA("SQLDA1","SQLN")=4
00180 END
00190 SQLDA("SQLDA1","SQLD")=4
00200 EXEC_SQL:| PREPARE S1 FROM: SQL_TEXT
00210 END
00220 EXEC_SQL:| DESCRIBE S1 INTO SQLDA1
00230 END
00240 EXEC_SQL:| DECLARE C1 CURSOR FOR S1
00250 END
00260 EXEC_SQL:| OPEN C1
00270 END
00280 EXEC_SQL:| FETCH C1 USING DESCRIPTOR SQLDA1
00290 END
00300 EMPL_NAME=SQLDA("SQLDA1","SQLDATA",1)
00310 EMPL_STREET=SQLDA("SQLDA1","SQLDATA",2)
00320 EMPL_STATE=SQLDA("SQLDA1","SQLDATA",3)
00330 EMPL_ZIP_CODE=SQLDA("SQLDA1","SQLDATA",4)
00340 SQLDA("SQLDA1")=QUIT

```

Deallocate an SQLDA

SQLDA(sqlda_name) = QUIT

sqlda_name

Description *Required.* Specifies the name of the SQLDA to be deallocated.

Format Must be a text literal or expression of 1-18 characters.

Considerations

- ◆ Naming conventions for SQL entities must be followed. Refer to the appropriate SQL language manual for the SQL database in use.
- ◆ Must be a previously defined SQLDA, via SQLDA(sqlदानame) = NEW.

Example

The following example shows how to deallocate an SQLDA. Line 340 deallocates an SQLDA named "SQLDA1".

```

00100 TEXT SQL_TEXT(254)
00110 TEXT EMPL_NAME(30), EMPL_STREET(30), EMPL_STATE(2)
00120 BIG EMPL_ZIP_CODE
00130 SQL_TEXT="SELECT NAME, STREET, STATE, ZIPCODE"
00140 SQL_TEXT=SQL_TEXT+" FROM EMPLOYEE.TABLE"
00150 SQLDA("SQLDA1")=NEW
00160 IF SQLDA("SQLDA1", "SQLN")<4
00170 .SQLDA("SQLDA1", "SQLN")=4
00180 END
00190 SQLDA("SQLDA1", "SQLD")=4
00200 EXEC_SQL:| PREPARE S1 FROM: SQL_TEXT
00210 END
00220 EXEC_SQL:| DESCRIBE S1 INTO SQLDA1
00230 END
00240 EXEC_SQL:| DECLARE C1 CURSOR FOR S1
00250 END
00260 EXEC_SQL:| OPEN C1
00270 END
00280 EXEC_SQL:| FETCH C1 USING DESCRIPTOR SQLDA1
00290 END
00300 EMPL_NAME=SQLDA("SQLDA1", "SQLDATA", 1)
00310 EMPL_STREET=SQLDA("SQLDA1", "SQLDATA", 2)
00320 EMPL_STATE=SQLDA("SQLDA1", "SQLDATA", 3)
00330 EMPL_ZIP_CODE=SQLDA("SQLDA1", "SQLDATA", 4)
00340 SQLDA("SQLDA1")=QUIT

```

Set SQLDA header information

SQLDA(sqlda_name,sqlda_header_element)=expression

sqlda_name

Description *Required.* Specifies the name of the SQLDA to be accessed.

Format Must be a text literal or expression of 1-18 characters.

Considerations

- ◆ Naming conventions for SQL entities must be followed. Refer to the appropriate SQL language manual for the SQL database in use.
- ◆ Must be a previously defined SQLDA, via SQLDA(sqlaname) = NEW.

sqlda_header_element

Description *Required.* Specifies the SQLDA header element that is accessed.

Format Must be a text literal or an expression that evaluates to one of the SQLDA header element names shown in the table following the next parameter description.

expression

Description *Required.* Specifies the data to be transferred from the MANTIS program into the SQLDA.

Format Must be consistent with the data type of the SQLDA element being stored (either text or numeric).

Consideration Certain SQLDA elements are read-only and cannot have data from the MANTIS program stored in them. In some cases, storing data in one SQLDA element causes MANTIS to automatically update other SQLDA elements.

Example

In the following example, SQLDA header elements are set in lines 250 and 270:

```
00100 TEXT SQL_TEXT(254)
00110 TEXT EMPL_NAME(30),EMPL_STREET(30),EMPL_STATE(2)
00120 BIG EMPL_ZIP_CODE
00130 EMPL_NAME="JONES"
00140 EMPL_STREET=" NEW STREET ADDRESS"
00150 EMPL_STATE="OH"
00160 EMPL_ZIP_CODE="12345"
00170 SQL_TEXT="UPDATE EMPLOYEE.TABLE SET"
00180 SQL_TEXT=SQL_TEXT+" NAME = ?, STREET = ?,"
00190 SQL_TEXT=SQL_TEXT+" STATE = ?, ZIP_CODE = ?"
00200 SQL_TEXT=SQL_TEXT+" WHERE NAME = ?"
00210 EXEC_SQL:| PREPARE S1 FROM :SQL_TEXT
00220 END
00230 SQLDA ("SQLDA1")=NEW
00240 IF SQLDA("SQLDA1", "SQLN")<5
00250 .SQLDA("SQLDA1", "SQLN")=5
00260 END
00270 SQLDA("SQLDA1", "SQLD")=5
00280 SQLDA("SQLDA1", "SQLDATA", 1)=EMPL_NAME
00290 SQLDA("SQLDA1", "SQLDATA", 2)=EMPL_STREET
00300 SQLDA("SQLDA1", "SQLDATA", 3)=EMPL_STATE
00310 SQLDA("SQLDA1", "SQLDATA", 4)=EMPL_ZIP_CODE
00320 SQLDA("SQLDA1", "SQLDATA", 5)=EMPL_NAME
00330 EXEC_SQL:| EXECUTE S1 USING DESCRIPTOR SQLDA1
00340 END
00350 SQLDA("SQLDA1")=QUIT
```

The following table lists and describes the SQLDA header elements:

Header element name	MANTIS compatible data type	Contents / considerations	Updateable?
SQLDAID	TEXT(8)	Eyecatcher. Set by SQL.	No
SQLDABC	BIG	SQLDA length. Set by SQL when the SQLDA is allocated; modified when SQLN is changed.	No
SQLN	BIG	Number of repeating groups (SQLVAR) in the SQLDA. Set using installation defined default value when the SQLDA is allocated. Can be modified by the MANTIS program if needed.	Yes
SQLD	BIG	Number of repeating groups currently in use. Set by SQL as the result of a DESCRIBE, can be set by program when necessary.	Yes

Move data into an SQLDA repeating group

SQLDA(sqlda_name,repeating_element,index)=expression

sqlda_name

- Description** *Required.* Specifies the name of the SQLDA to be accessed.
- Format** Must be a text literal or expression of 1-18 characters.
- Consideration** Naming conventions for SQL entities must be followed. Refer to the appropriate SQL language manual for the SQL database in use.
-

sqlda_repeating_element

- Description** *Required.* Specifies the repeating element of the SQLDA that is accessed.
- Format** Must be a text literal or expression that evaluates to one of the SQLDA repeating element names shown in the table following the next parameter description.
-

index

- Description** *Required* when accessing repeating elements. Specifies the group of SQLDA repeating elements that is accessed.
- Format** Must be a numeric literal or expression not less than one and not greater than the maximum number of repeating groups currently in the SQLDA (SQLN), inclusive.

General consideration

Using certain SQLDA elements causes other SQLDA elements to automatically be set by MANTIS. For example, storing data into the SQLDA element "SQLDATA" causes MANTIS to set the data type ("SQLTYPE") and data length ("SQLLEN") elements automatically. Storing data into the SQLDA "SQLIND" element updates the data type element ("SQLTYPE") to show that an indicator variable is present.

Example

In the following example, lines 280 - 320 store data from the MANTIS program into SQLDA repeating group elements.

```

00100 TEXT SQL_TEXT(254)
00110 TEXT EMPL_NAME(30),EMPL_STREET(30),EMPL_STATE(2)
00120 BIG EMPL_ZIP_CODE
00130 EMPL_NAME="JONES"
00140 EMPL_STREET=" NEW STREET ADDRESS"
00150 EMPL_STATE="OH"
00160 EMPL_ZIP_CODE="12345"
00170 SQL_TEXT="UPDATE EMPLOYEE.TABLE SET"
00180 SQL_TEXT=SQL_TEXT+" NAME = ?, STREET = ?, "
00190 SQL_TEXT=SQL_TEXT+" STATE = ?, ZIP_CODE = ?"
00200 SQL_TEXT=SQL_TEXT+" WHERE NAME = ?"
00210 EXEC_SQL:| PREPARE S1 FROM :SQL_TEXT
00220 END
00230 SQLDA ("SQLDA1")=NEW
00240 IF SQLDA("SQLDA1","SQLN")<5
00250 .SQLDA("SQLDA1","SQLN")=5
00260 END
00270 SQLDA("SQLDA1","SQLD")=5
00280 SQLDA("SQLDA1","SQLDATA",1)=EMPL_NAME
00290 SQLDA("SQLDA1","SQLDATA",2)=EMPL_STREET
00300 SQLDA("SQLDA1","SQLDATA",3)=EMPL_STATE
00310 SQLDA("SQLDA1","SQLDATA",4)=EMPL_ZIP_CODE
00320 SQLDA("SQLDA1","SQLDATA",5)=EMPL_NAME
00330 EXEC_SQL:| EXECUTE S1 USING DESCRIPTOR SQLDA1
00340 END
00350 SQLDA("SQLDA1")=QUIT

```

Repeating element name	MANTIS compatible data type	Contents / considerations	Updateable?
SQLTYPE	BIG	SQL data type code. Code differs depending on whether set by SQLDATA or SQLIND. Set when the value of the variable is transferred via SQLDATA or SQLIND.	No
SQLLEN	BIG	Length of data element in current repeating group. Set when value of variable is transferred via SQLDATA.	No
SQLDATA	TEXT, BIG, or KANJI	Subfunction that moves data, sets SQLTYPE and SQLLEN, and sets address of the data in the SQLDA. Used to transfer value of variable between database and MANTIS.	Yes
SQLIND	BIG	Subfunction that moves data, sets SQLTYPE and address of the data in the SQLDA. Used to transfer value of the indicator variable between database and MANTIS.	Yes
SQLNAME	TEXT(30)	SQL column name. Set by SQL, can be reset by the MANTIS program.	Yes

Read header elements

SQLDA(sqlda_name,sqlda_header_element)

sqlda_name

Description *Required.* Specifies the name of the SQLDA to be accessed.

Format Must be a text literal or expression of 1-18 characters.

Consideration Naming conventions for SQL entities must be followed. Refer to the appropriate SQL language manual for the SQL database in use.

sqlda_header_element

Description *Required.* Specifies the SQLDA header element that is accessed.

Format Must be a text literal or an expression that evaluates to one of the SQLDA header element names shown in the table under “[Set SQLDA header information](#)” on page 61.

Example

In the following example, line 160 shows the SQLDA "SQLN" header element being read from the SQLDA.

```
00100 TEXT SQL_TEXT(254)
00110 TEXT EMPL_NAME(30),EMPL_STREET(30),EMPL_STATE(2)
00120 BIG EMPL_ZIP_CODE
00130 SQL_TEXT="SELECT NAME, STREET, STATE, ZIPCODE"
00140 SQL_TEXT=SQL_TEXT+" FROM EMPLOYEE.TABLE"
00150 SQLDA("SQLDA1")=NEW
00160 IF SQLDA("SQLDA1","SQLN")<4
00170 .SQLDA("SQLDA1","SQLN")=4
00180 END
00190 SQLDA("SQLDA1","SQLD")=4
00200 EXEC_SQL:| PREPARE S1 FROM: SQL_TEXT
00210 END
00220 EXEC_SQL:| DESCRIBE S1 INTO SQLDA1
00230 END
00240 EXEC_SQL:| DECLARE C1 CURSOR FOR S1
00250 END
00260 EXEC_SQL:| OPEN C1
00270 END
00280 EXEC_SQL:| FETCH C1 USING DESCRIPTOR SQLDA1
00290 END
00300 EXEC_SQL:| EXECUTE S2 USING DESCRIPTOR SQLDA1
00310 END
00320 EMPL_NAME=SQLDA("SQLDA1","SQLDATA",1)
00330 EMPL_STREET=SQLDA("SQLDA1","SQLDATA",2)
00340 EMPL_STATE=SQLDA("SQLDA1","SQLDATA",3)
00350 EMPL_ZIP_CODE=SQLDA("SQLDA1","SQLDATA",4)
00360 SQLDA("SQLDA1")=QUIT
```

Move data from an SQLDA repeating group into a MANTIS program

SQLDA(sqlda_name,sqlda_repeating_element,index)

sqlda_name

- Description** *Required.* Specifies the name of the SQLDA to be accessed.
- Format** Must be a text literal or expression of 1-18 characters.
- Consideration** Naming conventions for SQL entities must be followed. Refer to the appropriate SQL language manual for the SQL database in use.
-

sqlda_repeating_element

- Description** *Required.* Specifies the repeating element of the SQLDA that is accessed.
- Format** Must be a text literal or expression that evaluates to one of the SQLDA repeating element names (see the table under [“Move data into an SQLDA repeating group”](#) on page 64).
-

index

- Description** *Required* when accessing repeating elements. Specifies the group of SQLDA repeating elements that is accessed.
- Format** Must be a numeric literal or expression between one and the maximum number of repeating groups currently in the SQLDA (SQLN), inclusive.

Example

In the following example, lines 300 - 330 show SQLDA repeating elements being moved from the SQLDA into the MANTIS program.

```
00100 TEXT SQL_TEXT(254)
00110 TEXT EMPL_NAME(30),EMPL_STREET(30),EMPL_STATE(2)
00120 BIG EMPL_ZIP_CODE
00130 SQL_TEXT="SELECT NAME, STREET, STATE, ZIPCODE"
00140 SQL_TEXT=SQL_TEXT+" FROM EMPLOYEE.TABLE"
00150 SQLDA("SQLDA1")=NEW
00160 IF SQLDA("SQLDA1","SQLN")<4
00170 .SQLDA("SQLDA1","SQLN")=4
00180 END
00190 SQLDA("SQLDA1","SQLD")=4
00200 EXEC_SQL:| PREPARE S1 FROM: SQL_TEXT
00210 END
00220 EXEC_SQL:| DESCRIBE S1 INTO SQLDA1
00230 END
00240 EXEC_SQL:| DECLARE C1 CURSOR FOR S1
00250 END
00260 EXEC_SQL:| OPEN C1
00270 END
00280 EXEC_SQL:| FETCH C1 USING DESCRIPTOR SQLDA1
00290 END
00300 EMPL_NAME=SQLDA("SQLDA1","SQLDATA",1)
00310 EMPL_STREET=SQLDA("SQLDA1","SQLDATA",2)
00320 EMPL_STATE=SQLDA("SQLDA1","SQLDATA",3)
00330 EMPL_ZIP_CODE=SQLDA("SQLDA1","SQLDATA",4)
00340 SQLDA("SQLDA1")=QUIT
```

The following table shows the mapping between SQL data types and MANTIS data types. Use this table to determine what field types DB2 is returning for a dynamic statement via an SQLDA. You can then assign the returned fields to a variable of compatible type.

SQL data type	Description	SQL type* (output host variables)	SQL type set by MANTIS (input host variables)	MANTIS type
DATE	Calendar Date	384/385	448/449	TEXT
TIME	Time	388/389	448/449	TEXT
TIMESTAMP	Timestamp	392/393	448/449	TEXT
VARCHAR	Variable String	448/449	448/449	TEXT
CHAR	Fixed length string	452/453	448/449	TEXT
LONG VARCHAR	Long variable string	456/457	448/449	TEXT
VARGRAPHIC	Variable graphic string	464/465	464/465	KANJI
GRAPHIC	Fixed length graphic string	468/469	464/465	KANJI
LONG VARGRAPHIC	Long variable graphic string	472/473	464/465	KANJI
FLOAT	Floating point number	480/481	480/481	BIG
DECIMAL	Packed decimal number	484/485	480/481	BIG
INTEGER	Long integer	496/497	480/481	BIG
SMALLINT	Short integer	500/501	480/481	BIG

* The first number in the column is used when no indicator variable is present. This value does not allow NULL values. The second number is used when an indicator variable is present, and NULL data values can be specified.

SQL statements larger than 254 characters

To dynamically prepare an SQL statement for execution, use the SQL PREPARE statement. The SQL statement to be executed is contained either in the PREPARE statement as a text literal, or in the host variable specified in the PREPARE statement.

Because the SQL statement is text, the host variable specified in the PREPARE statement must be a MANTIS TEXT variable. MANTIS TEXT variables are limited to 254 characters, which can cause a problem for large SQL statements.

MANTIS SQL Support allows large SQL statements to be placed in a MANTIS TEXT array. MANTIS SQL Support combines the contents of all rows in a MANTIS TEXT array into a single SQL statement. This combined text is then used as the SQL statement text in the SQL PREPARE statement.

In order to use this feature, the host variable specified in the PREPARE statement:

- ◆ Must be declared as a MANTIS TEXT array.
- ◆ Must not have subscripts specified. If subscripts are present, MANTIS uses only the text of the subscripted row as the SQL statement.

When these conditions are met, MANTIS uses the contents of the entire array as the SQL statement text. The text is combined “as is” from all rows in the array. No characters are added or deleted so that SQL text may split row boundaries if required. Rows containing no data are ignored.



Because ALL rows of the array are combined, some rows may need to be cleared from one SQL statement execution to another. If they are not cleared, an SQL statement may be combined with text remaining from a previous SQL statement. Executing this statement may cause errors that are difficult to diagnose.

An example of how to use a MANTIS TEXT array to contain a large SQL statement is shown below:

```

40 ..TEXT SQL_TEXT(10,20)
50 ..SQL_TEXT(1)="SELECT EMPLNO, EMPLNM"
60 ..SQL_TEXT(2)="M"
70 ..SQL_TEXT(5)="FROM EMPLOYEE.TABLE"
80 ..SQL_TEXT(6)="WHERE EMPLNO= ?"
90 ..EXEC_SQL
100 ...| PREPARE S1 FROM :SQL_TEXT
110 ..END
120 ..CLEAR SQL_TEXT

```

Array can be any length/number.

Text can split array boundaries.

Rows 3 & 4 ignored-contains no data.

No spaces are added or deleted.

No subscripts specified.

Clear for future use.

The SQL text used by the SQL PREPARE statement in this example is:

```
SELECT EMPLNO, EMPLNM FROM EMPLOYEE.TABLE WHERE EMPLNO= ?
```


5

Preparing MANTIS SQL support programs for static execution mode (DB2 for OS/390)

Static execution mode vs. dynamic execution mode

Static execution mode

Static execution mode allows SQL statements in MANTIS SQL programs to be precompiled in a way similar to SQL in COBOL. The SQL statements from a MANTIS program are extracted into a BAL (Basic Assembler Language) source module. The BAL source module is then precompiled and assembled as is any other program in SQL in COBOL. When the MANTIS SQL program executes in static execution mode, the SQL statements are executed from the precompiled BAL module rather than by dynamic SQL statements. This process allows MANTIS to execute SQL statements with performance similar to SQL in COBOL.

In MANTIS SQL Support for DB2, executing the program in static execution mode instead of dynamic execution mode uses fewer resources.

Dynamic execution mode

In dynamic execution mode, dynamic SQL statements execute all SQL statements in your MANTIS program, eliminating the need for an SQL precompile step each time you change the program. You can code SQL statements in a MANTIS program and immediately RUN the program. When errors are detected, you can modify these statements and immediately re-execute the program. The flexibility of dynamic execution mode requires considerable computer resources. While using these resources is desirable during program development, they should be limited when the program is ready for production.

Preparing your programs to run in static execution mode

The five steps required to prepare your programs to run in static execution mode are:

1. SQL binding the MANTIS program.

Before you can execute a MANTIS SQL program in static execution mode, you must SQL bind it. (Do not confuse SQL binding with HPO (High Performance Option) binding. Both HPO bound and unbound programs can be SQL bound.) MANTIS SQL programs are SQL bound using the SQL Bind Facility (see “[SQL binding the MANTIS program](#)” on page 82). When you SQL Bind the MANTIS program, MANTIS extracts information about each SQL statement in the MANTIS program and places it on the MANTIS cluster. This information is in a file called the SQL Bind Information. After MANTIS creates the SQL Bind Information, MANTIS marks the program as SQL bound and terminates the SQL Bind Facility. You must perform all remaining steps outside of MANTIS.

2. Generating the SQL Support Source Module.

After the SQL bind is complete, a batch utility program reads the SQL Bind Information output from the SQL Bind and generates a BAL source code module containing the SQL statements from the MANTIS program. The BAL source module is called the SQL Support Source Module.

3. Creating the SQL Support Load Module.

You then precompile the SQL Support Source Module using the DB2 precompiler for BAL programs. This creates the DB2 Database Request Module (DBRM) and expanded BAL source code for the SQL Support Source Module. Assemble (compile) this expanded source code to create the SQL Support Load Module.

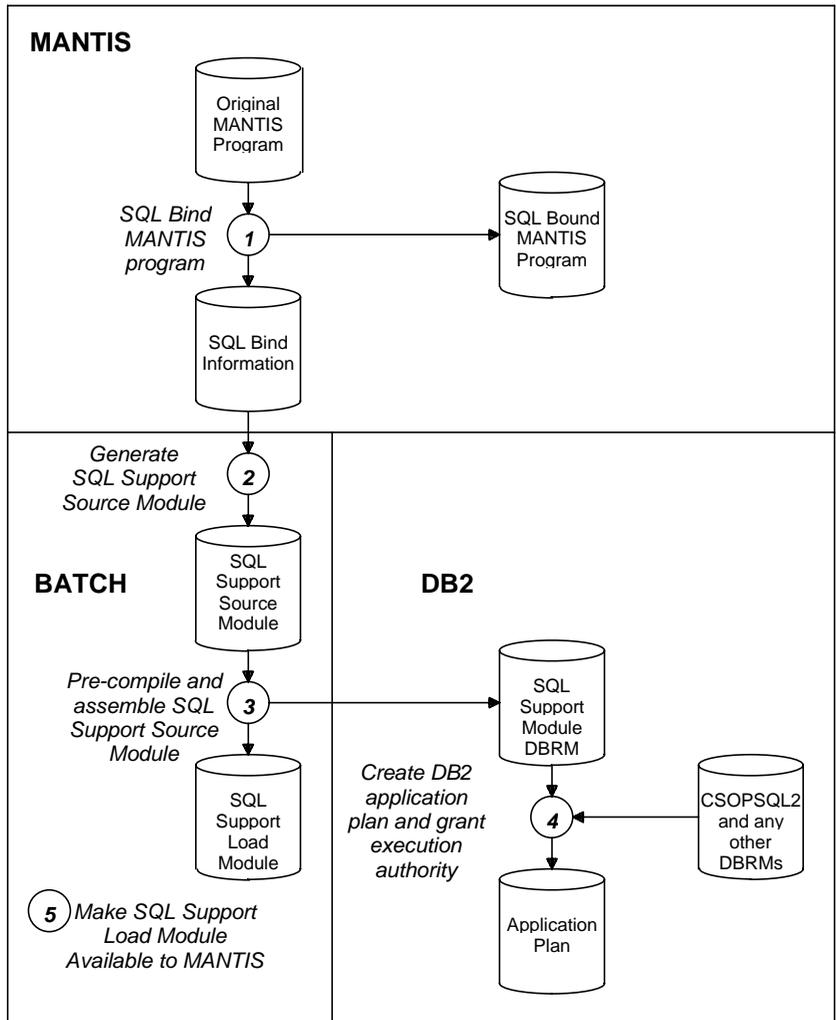
4. Creating the DB2 Application Plan and granting execution authority.

If necessary, combine the DB2 DBRM created in the previous step with other DBRMs to create the DB2 Application Plan for the MANTIS program. Use the DB2 BIND command (do not confuse this with the MANTIS HPO BIND or SQL BIND commands). Use the DB2 GRANT command to give appropriate execution authority to this Application Plan.

5. Making the SQL Support Load Module available to MANTIS.

Make the SQL Support Module available so MANTIS can execute it when needed through the MANTIS program executing in static execution mode. Include (linkedit) the SQL Support Load Module with MANTIS, or link it alone to the library so that MANTIS can dynamically load the support module when needed.

For details on the preceding five steps, see “[Preparing a program to run in static mode](#)” on page 82. The following figure illustrates the five steps required in order to run MANTIS SQL programs in static execution mode:



Performance and programming considerations

Programs executing in static execution mode cannot always execute exactly as they do in dynamic execution mode. This section lists certain differences and restrictions for programs executing in static execution mode. Review these differences before preparing your program for execution in static mode.

Determining host variable data types

When a MANTIS program is SQL bound, MANTIS temporarily HPO binds the program (if it is not already HPO bound). This binding determines the data type and length of MANTIS variables used as host variables in SQL statements. This information must be known so that you can perform the precompile and assembly. MANTIS considers any host variables that are not resolved by the HPO bind to be undefined and may default to BIG, depending on the option you specify when you execute the SQL Bind option (see “[SQL binding the MANTIS program](#)” on page 82).

Because HPO binding determines host variable data types, structure the MANTIS program as if it were to be HPO bound. Define all MANTIS variables that are used as host variables in the program before any statements that cause HPO binding to terminate. This is especially true for nonnumeric variables (TEXT and KANJI). For more information on HPO binding, refer to [MANTIS Facilities, OS/390, VSE/ESA](#), P39-5001. If a MANTIS TEXT variable defaults to BIG in the SQL Support Load Module, the program will not run in static execution mode. Either the SQL database or MANTIS will return an error, depending on how the text variable is used.

SQL indicator variables

In dynamic execution mode, when an SQL statement is executed, DB2 may return indicator variables. If so, MANTIS allocates space for them and permits the SQL statement to execute. However, MANTIS only returns the value of the indicator variable when the indicator variable is present in the SQL statement in the MANTIS program.

If a program is prepared for static execution mode, however, MANTIS cannot determine whether indicator variables will be returned before the program executes. An SQL statement requiring an indicator variable fails unless the indicator variable is defined in the MANTIS program. To avoid this failure in the SQL support module, specify indicator variables in your SQL statements wherever DB2 requires them.

Declaring SQL cursors multiple times

In dynamic execution mode, you can declare an SQL cursor multiple times in the MANTIS program, provided you execute only one of the DECLARE statements. MANTIS SQL Support does not return an error until your program executes the second DECLARE statement for an existing SQL cursor.

In static execution mode, you cannot declare an SQL cursor more than once. When the SQL Support source module is input to the SQL precompiler program, the precompiler generates an error for any SQL cursor defined more than once, regardless of whether the declaration is ever executed.

The following example shows an SQL cursor declared multiple times in a MANTIS SQL program. It executes successfully in dynamic execution mode, but receives an error when you precompile the SQL Support Module:

```
EXEC_SQL
.| DECLARE C1 CURSOR FOR SELECT ...
END
IF ZERO=TRUE
.EXEC_SQL
..| DECLARE C1 CURSOR FOR SELECT ...
.END
END
```

ENTRY statement parameters as host variables

HPO binding is done during the SQL binding. Therefore, do not use parameters that are passed on an ENTRY statement as host variables in SQL statements in static execution mode. All ENTRY statement parameters default to a data type of BIG because their actual data types cannot be determined until the program is executed. In dynamic execution mode, parameters passed in an ENTRY statement can be used as host variables in SQL statements because the data types of the parameters are known when the SQL statement is executed.

If you must use parameters passed on an ENTRY statement as host variables in an SQL statement, explicitly define separate MANTIS variables for the passed parameters; move the parameters to MANTIS variables and use the explicitly defined variables as host variables in the SQL statement text. This is shown in the following example. The ENTRY statement parameters (X, Y, Z) have duplicate explicitly defined MANTIS variables (Q, R, S) that are used as host variables in the SQL statement text.

ENTRY ABC(X, Y, Z)	
.TEXT Q(10):BIG R	<i>Explicitly define all host variables.</i>
.SMALL S	
.S=Z	<i>Set value for input host variable.</i>
.EXEC_SQL: SELECT * INTO :Q, :R FROM SQLTABLE	
. . WHERE SQLCOLUMN = :S	<i>Binding stops at this statement.</i>
.END	
.X=Q	<i>Return values for output host variables.</i>
.Y=R	
EXIT	

The example above executes successfully in either dynamic or static execution mode. The following example will execute successfully in dynamic execution mode but may cause errors in static execution mode.

```
ENTRY ABC(X, Y, Z)
.EXEC_SQL
.. | SELECT * INTO :X, :Y FROM SQLTABLE
.. | WHERE SQLCOLUMN> :Z
.END
EXIT
```

Multiple row result sets with SELECT

In MANTIS dynamic execution mode, a SELECT statement always returns the first row of the result set regardless of the number of rows in the result set. In static execution mode, the SQL database returns an error if the result set for the SELECT contains more than one row. In the example below, the first row in the result set built by the SELECT is always returned, even if the result set contains more than one row. This example causes an SQL error to occur in static execution mode if the result set built by the SELECT contains multiple rows.

```
EXEC_SQL
.| SELECT * INTO :X, :Y FROM SQLTABLE
.| WHERE SQLCOLUMN > :Z
END
```

You can avoid this problem by using SQL cursors instead of the SELECT statement in static execution mode. The following example is the equivalent of the previous example using SQL cursors instead of SELECT.

```
EXEC_SQL
.| DECLARE C1 CURSOR FOR SELECT *
.| FROM SQLTABLE WHERE SQLCOLUMN > :Z
END
.|
EXEC_SQL
.| OPEN C1
END
EXEC_SQL
.| FETCH C1 INTO :X, :Y
END
EXEC_SQL
.| CLOSE C1
END
```

Binding on the target system

It is necessary to SQL bind the MANTIS program on the system where it will be executed or a clone of that system because SQL binding uses HPO binding to determine host variable data types. If MANTIS variables that are used as host variables are defined in HPO bound entities (screens, files, etc.), and these entities are different on the target system from those on the system where the SQL bind is done, unpredictable results, errors, or abends may occur.

Preparing a program to run in static mode

The steps required to prepare a program for static mode execution are:

1. SQL binding the MANTIS program
2. Generating the SQL Support Source Module
3. Pre-compiling and assembling the SQL Support Source Module
4. Creating the DB2 Application Plan and granting execution authority
5. Making the SQL Support Load Module available to MANTIS



Modifying anything created by these steps may result in inconsistent entities that may prevent your program from being executed in static execution mode.

SQL binding the MANTIS program

The first step in preparing a MANTIS program to run in static execution mode is to SQL bind the program using the MANTIS SQL Bind facility. Access the Bind Options for SQL programs from the MANTIS Program Design Facility menu shown in the screen illustration under “[SQL binding a MANTIS program online](#)” on page 83. The options include binding, unbinding, checking, and maintenance actions. If you are unfamiliar with the Program Design Facility or the Bind Options, refer to *MANTIS Program Design and Editing, OS/390, VSE/ESA*, P39-5013, for details.

SQL binding a MANTIS program online

To SQL bind a MANTIS program, select the SQL Bind option at the Program Design Facility menu as shown in the following illustration and press ENTER:

```

PRGMMENU01      Program Design Facility (TEST)      YYYY/MM/DD HH:MM:SS
====>

Please select one of the menu items below.

      Program      Component Engineering  Bind Options      Utilities
16  1. List        7. CEF Check      12. HPO Check     18. Audit Trail
    2. Edit        8. " Compose      13. " Bind        19. Browse Audit Trail
    3. Profile     9. " Decompose    14. " Unbind      20. " Prgm Profile
    4. Purge       10. CREF Programs 15. SQL Check     21. Trigger List
    5. Copy        11. Bill of Materials 16. " Bind        22. SQL Maint
    6. Rename      17. " Unbind

```

000: READY
F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F9=RETRIEVE F12=CANCEL ...

The next panel displayed is the SQLBIND Program Entry panel (see the following screen illustration). This panel allows you to specify the program or programs you want to SQL bind. You can specify a single program name, a range of program names, or a generic pattern of names. You can also specify Entry Options, Function Options and Process Statistics.

The Function Options for the SQLBIND Program Entry panel are described below. For information on the Entry Options or Process Statistics, refer to *MANTIS Facilities, OS/390, VSE/ESA*, P39-5001. In this example, a single program name is specified and the defaults for all other options are accepted. Press PF6 to execute.

```

PRGMENT101E   SQLBIND Program Entry                               YYYY/MM/DD HH:MM:SS
====>
From
  Library . . . . TEST
  Name . . . . SQL_DB2
  Description .
  Password :
  :

Thru
  Name . . . .

Entry Options      Function Options      Process Statistics
Immediate? . . . . Y      Preprocess Options:      Processed . .
Confirmation? . . . Y      . .Save/Replace . .     S Skipped . . .
Addendum? . . . . N      . .Keep/Revoke . .     K Errors . . .
                          . .Block/Noblock . .     B
                          Undefined Variables?      1

000: READY
F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F6=EXECUTE F7=CONFIRM ...
    
```

PREPROCESS OPTIONS:

Description *Required.* Specifies the options to be used for the SQL bind.

SAVE/REPLACE

Description *Required.* Specifies whether the SQL Bind Information is to be created ("S"), or if existing SQL Bind Information for the program is to be replaced ("R").

Default SAVE ("S")

Considerations

- ◆ An error will be returned if "S" is specified for a program that has been previously SQL bound.
 - ◆ For users of earlier releases of MANTIS SQL Support, this option replaces the REBIND option.
-

KEEP/REVOKE

Description *Ignored.* Applies only to programs using DB2 for VSE and VM.

BLOCK/NOBLOCK

Description *Ignored.* Applies only to programs using DB2 for VSE and VM.

UNDEFINED VARIABLES?

Description	<i>Required.</i> Specifies what should be done if undefined variables are encountered during the SQL Bind.
Default	1
Options	<ol style="list-style-type: none">1 QUIT WITH DISPLAY. If any undefined variables are found, MANTIS displays the undefined variables with their line numbers and does not SQL bind the program. These variables must be changed in the MANTIS program or another option must be selected. If no undefined variables are found, the program is SQL bound.2 CONTINUE NO DISPLAY. Binding continues even if undefined variables are encountered. All undefined variables are assigned a data type of BIG, but these variables are not displayed.3 CONTINUE WITH DISPLAY. Binding continues even if MANTIS encounters undefined variables. Undefined variables are assigned a data type of BIG and displayed with their line numbers.

Considerations

- ◆ Nonnumeric variables that default to BIG can create inconsistencies that prevent static mode execution (see “[Declaring SQL cursors multiple times](#)” on page 79).
- ◆ Any undefined variables displayed include the line number on which they appear. Undefined variables are not displayed with option 2.
- ◆ If you set SAVE/REPLACE=“R” and the SQL bind fails, any existing SQL Bind Information records for the named SQL Support Module are deleted from the MANTIS Cluster. The program will remain marked as SQL bound and the Support Module created earlier is still valid. If the SQL program was previously HPO bound, it remains HPO bound.

When you press PF6 to execute, MANTIS returns the description of the program you have specified for binding and asks you to either CONFIRM or SKIP the action as shown in the following screen illustration. Press PF7 to confirm the action.

```

PRGMENT101E      SQLBIND Program Entry      YYYY/MM/DD HH:MM:SS
====>  _-
From
  Library . . . . TEST
  Name . . . . SQL_DB2      Password :           :
  Description . DYNAMIC SQL DB2 ACCESS PROGRAM

Thru
  Name . . . .

Entry Options      Function Options      Process Statistics
Immediate? . . . . Y      Preprocess Options:      Processed . .
Confirmation? . . . . Y      . .Save/Replace . . . . S      Skipped . . . .
Addendum? . . . . N      . .Keep/Revoke . . . .      Errors . . . .
                          . .Block/Noblock . .
                          Undefined Variables? 1

SQLBIND      SQL_DB2

U01: CONFIRM OR SKIP
F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F6=EXECUTE F7=CONFIRM ...

```

The next panel displayed is the SQL Bind Information panel (see the following screen illustration). The options for this panel are described below. After making any changes or specifications necessary, press PF6 to complete the binding process. If no errors are encountered, a confirmation message, NUCQBAI: SQL BIND COMPLETED SUCCESSFULLY, will be returned. Press PA2 to return to the Program Design Facility menu. Press PF3 to exit and return to the MANTIS Facility Selection menu.

```
SQLBIND                SQL BIND INFORMATION PANEL

ACTION..... : SQLBIND  :
MANTIS PROGRAM NAME.. : SQL_DB2      :
SQL BINDING TYPE..... : STATIC      :
SQL STATEMENTS.....  : 5           :

BOUND BY..... :           :
BOUND AT..... :           :
MANTIS RELEASE..... :           :

SQL MODULE NAME ..... : SQLMOD3  :
SQL MODULE OWNER NAME :           :
SQL OWNER PASSWORD    :           :
SQL DATA BASE TYPE .. : DB2     :
```

ENTER=CONTINUE PF3=EXIT PF6=EXECUTE PA2=CANCEL

MANTIS PROGRAM NAME

Description *Protected.* Displays your user name and the name of the MANTIS SQL program you are binding.

SQL BINDING TYPE

Description *Protected.* Displays the type of SQL bind that will be performed (STATIC).

SQL STATEMENTS

Description *Protected.* Displays the number of SQL statements in the program you are binding.

SQL MODULE NAME

Description *Required.* The name of the SQL Support Module. If the SQL program was previously bound, the name of the previously created SQL Support Module is displayed; otherwise, this field is blank.

Default The existing SQL Support Module name (if previously bound).

Format 1-7 alphanumeric character symbolic name. The first character may be A-Z, @, #, \$; characters 2-7 may be these characters plus 0-9.

Consideration The name must be unique to the SQL database and to MANTIS.

SQL MODULE OWNER NAME

Description *Ignored.* Applies only to programs using DB2 for VSE and VM.

SQL OWNER PASSWORD

Description *Ignored.* Applies only to programs using DB2 for VSE and VM.

SQL DATA BASE TYPE

Description *Protected.* The database for which the bind will be done.

Default DB2

Consideration DB2 for OS/390 is currently the only database that supports static execution mode.

SQL binding a MANTIS program with batch MANTIS

You can SQL bind one or more SQL programs by using Batch MANTIS. Sample input for SQL binding with Batch MANTIS is in the following screen illustration. Sample JCL to SQL bind MANTIS programs using Batch MANTIS is supplied with the MANTIS installation tape. Refer to the installation documentation or contact your system administrator for more information. Note that the asterisk (*) denotes three separate fields.

```
<ECHO=ON>user;password;
<FAULT=OFF>;
<BLANK=OFF>;1;
CONTROL:SQLBIND_FACILITY;

user:program1;name1;owner;preprocess_option*;when undefined_option;password1

user:program2;name2;owner;preprocess_option*;when undefined_option;password2
.
<PA2>
<PA2>
```

Field equivalents

name 1	SQL_module_name1
owner	SQL_module_owner
preprocess_option	preprocessing_option (3 fields)
password1	SQL_owner_password1
name 2	SQL_module_name2
password2	SQL_owner_password2

General considerations

- ◆ Because SQL Bind Information is added to the MANTIS cluster during an SQL bind, the MANTIS cluster must be updateable by Batch MANTIS.
- ◆ The output is routed to the printer, so you may want to place the PRINTER DD card ahead of the TERMINAL DD card or use TERMINAL DD DUMMY.
- ◆ If you process multiple programs that contain undefined variables in batch, the undefined variables are printed on the page before the panel containing their program name.

Code	Description
0	Successful SQL bind.
4	Warning on SQL bind operation. Successful SQL bind.
8	Error on SQL bind operation. Unsuccessful SQL bind.
16	Fatal error raised on SQL bind operation. Unsuccessful SQL bind and/or termination of execution.

Checking the results of an SQL bind

To see whether an SQL program is bound, select option 1 (Program List) from the Program Design Facility menu (see the screen illustration under “SQL binding a MANTIS program online” on page 83) and press ENTER. The next panel displayed is the Program Directory List shown in the following screen illustration. If the program has been bound, the BND field will contain an S, meaning that the program has been bound for static mode execution.

For more information on the Program Directory List and a description of the fields, refer to *MANTIS Program Design and Editing, OS/390, VSE/ESA*, P39-5013.

```

PRGMLIST01          Program Directory List                      YYYY/MM/DD HH:MM:SS
====>  _
From
Action      Name                      Date      Time      Ver  BND  Status
-----
_____ SQL_DB2                      92/02/22 09:15:37  5  S  ACTIVE
_____ SQL_PRGM1                     92/02/22 09:15:37  5  ACTIVE
_____ SQL_PRGM2                     92/02/22 09:15:37  5  ACTIVE
_____ SQL_PRGM3                     92/02/22 09:15:37  5  ACTIVE
_____ SQL_PRGM4                     92/02/22 09:15:37  5  ACTIVE
_____ SQL_PRGM5                     92/02/22 09:15:37  5  ACTIVE
_____ .
_____ .
_____ .

F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F8=FWD F9=RETRIEVE...
    
```

Generating the SQL support source module

After you SQL bind the MANTIS program, you must generate the SQL Support Source Module. This step requires the SQL Bind Information created during the SQL bind. Verify that the SQL bind information is available before proceeding (see “[Checking the results of an SQL bind](#)” on page 92).

A batch program, SQLGENR (not Batch MANTIS), creates the SQL Support Source Module. SQLGENER reads the SQL Bind Information from the MANTIS cluster and outputs the SQL Support Source Module as a BAL (Basic Assembler Language) source program.

For each execution of the batch SQL Support Source Module generate program, you can generate one or more SQL Support Source Modules. The SQL Support Source Module is placed in a designated OS/390 library. Sample input for the batch SQL Support Source Module generate program is shown in the following code sample. Sample JCL to execute the batch SQL Support Source Module generate program is included on the MANTIS installation tape. Refer to the installation documentation or see your MANTIS system administrator for more information.

```
//SYSIN DD *
USER(user:password)
DATA1(./ ADD NAME=#####)
DATA9(./ ENDUP)
DELETE(YES)
* THIS IS A COMMENT
support_module_name_1
support_module_name_2
support_module_name_3
support_module_name_4
/*
```

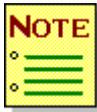
Considerations

- ◆ All input statements are placed in the OS/390 JCL SYSIN DD name. Only the USER (user:password) and one or more SQL Support Source Module names are required. USER(user:password) specifies the MANTIS user name and password used to perform the SQL bind. Processing stops if the user is omitted or unknown to MANTIS or if the password is invalid.
- ◆ The DATA1 statement inserts a statement at the beginning of each SQL Support Source Module generated. In the preceding code sample, the DATA1 statement inserts the string `“./ ADD NAME = source_module_name”` at the beginning of each module, using the module name from the MANTIS SQL Bind panel. DATA1 is optional; if not supplied, no string is inserted before each source module. DATA1() disables the insertion of the string specified in a previous DATA1 statement. DATA1() inserts a blank line. This statement is used with the IBM IEBUPDTE utility (see consideration 4).

You can enter up to 50 characters between the parentheses. To insert the source program's name in a statement, place exactly seven hash characters (#) in the desired location(s). No parentheses other than the two enclosing the string are allowed. In the preceding code sample, the DATA1 statement inserts `“./ ADD NAME=”` followed by the SQL Support Source Module name before each SQL Support Source Module generated.

- ◆ The DATA9 statement inserts a statement after the last SQL Support Source Module generated. No parentheses other than the two enclosing the string are allowed. You can enter up to 50 characters between the parentheses. No substitution of source program name for pound signs is permitted. DATA9() disables the insertion of the string specified in a previous DATA9 statement. DATA9() inserts a blank line. DATA9 is optional; if not supplied, no statement is inserted after the last support module. In the preceding code sample, the DATA9 statement inserts this string at the end of the file: `“./ ENDUP”`. This statement is used with the IBM IEBUPDTE utility (see consideration 4).

- ◆ The output of the SQL Support Source Module generate program is a single data set containing the BAL source code for all SQL Support Modules generated. The DATA1 and DATA9 parameters (see above) permit control statements to be inserted in the output dataset for use with the IBM IEBUPDTE utility program. When the SQL Support Source Module generate program terminates, the IEBUPDTE program can be executed to create individual SQL Support Source Modules from the output data set.
- ◆ When DELETE(YES) is specified, the SQL Bind Information is deleted from the MANTIS cluster if the SQL Support Source Module has been generated successfully. You can also set DELETE(NO). If DELETE(YES), a message is displayed indicating that SQL Bind Information was deleted. If omitted, DELETE defaults to YES.



If DELETE(YES), the MANTIS cluster must be updateable by the SQL Source Module generate program for the SQL Bind Information to be deleted.

General considerations

- ◆ Any statement that is not a USER, DATA1, DATA9, DELETE, or comment (*) must be the name of an SQL Support Module.
- ◆ Any statement beginning with an asterisk (*) in column 1 is a comment.
- ◆ Blank statements or leading blanks on any input statements are ignored.
- ◆ If an input statement contains an error, DELETE is set to NO when the error is detected. Unless a subsequent statement specifies DELETE(YES), the SQL Bind Information remains on the MANTIS cluster. In this case you need only rerun this step with corrected input statements.
- ◆ When input statements are repeated in the output listing, the password in the USER (user:password) is replaced with 16 asterisks.

If the SQL Support Source Module generation is successful, you receive a return code of zero and the following confirmation messages in the output listing:

```
SQLQGKI INPUT STATEMENT: sql_support_module_name
SQLQGLI SQL SUPPORT SOURCE MODULE GENERATION WAS SUCCESSFUL
SQLQJNI MODULE NAME .... sql_support_module_name
SQLQGOI PROGRAM NAME ... user:program_name
SQLQGUI SOURCE MODULE GENERATE UTILITY TERMINATED SUCCESSFULLY
```

You can also view the SQL Support Source Modules in the OS/390 libraries.

Creating the SQL support load module

After you generate the SQL Support Source Module, you must precompile it using the DB2 precompiler for BAL programs. You must assemble and linkedit the expanded source output from the DB2 precompiler with one of the DB2 attach facility modules (DSNCLI for CICS or DSNELI for TSO batch) to create the SQL Support Load Module.

Sample JCL to precompile, assemble, and linkedit SQL Support Source Modules is included on the MANTIS installation tape. The sample JCL includes the following:

- ◆ OS/390 JCL procedure (PROC) that executes:
 - DB2 precompiler
 - OS/390 BAL assembler
 - Linkage editor programs
- ◆ OS/390 JCL that executes the PROC

You can use either the sample JCL or your own, equivalent JCL to create the SQL Support Load Module.

General considerations

- ◆ Consult the MANTIS installation documentation or your system administrator for specific JCL to be executed.
- ◆ The sample JCL that executes the OS/390 PROC contains a "PROCLIB DD" statement. If your system does not support this statement, the sample PROC must be cataloged to a system procedure library or included with the JCL as an "in line" procedure.
- ◆ Before using the sample PROC, you must modify it. The PROC references several OS/390 libraries and must be updated with the names of those libraries on your system. The modifications necessary to the PROC are documented in the PROC. Refer to that documentation for more information.

- ◆ The sample JCL linkedits all CICS SQL Support Load Modules twice. The output of each linkedit is placed into a separate OS/390 library. This is done because the DB2 attach facility program is different for CICS (DSNCLI) than it is for batch (DSNELI). Because the SQL Support Load Module name is the same for both batch and online, separate OS/390 libraries are necessary. IMS SQL Support Load Modules are linkedited only once because the DB2 attach facility program (ASMTDLI) is the same for online and batch.
- ◆ If the MANTIS program that is SQL bound contains KANJI data in any of the SQL statements in the program, the DBCS option must be specified to the BAL assembler when the SQL Support Source Module is assembled. This is required because KANJI data in the MANTIS SQL program can cause BAL “G” type constants to be generated, which require the DBCS assembler option.

After precompiling, assembling, and linkediting the SQL Support Source Module, check the output listing for any error messages and verify that the SQL Support Load Module and DB2 Data Base Request Module (DBRM) were created.

Creating the DB2 application plan and granting execution authority

Before using an SQL Support Load Module, you must bind the DBRM created during the DB2 precompile into a DB2 Application Plan. The DB2 BIND command creates the DB2 Application Plan (do not confuse it with the MANTIS HPO or SQL BIND commands). One or more DB2 DBRMs can be bound into a DB2 Application Plan. These can include the DBRM used for dynamic execution mode (CSOPSQL2), if desired. After the Application Plan is created, use the DB2 GRANT command to provide the Application Plan with MANTIS authority.

You can perform the DB2 BIND and GRANT interactively using the DB2I facility or via a batch job. Sample input for the batch method is included on the MANTIS installation tape. Refer to the MANTIS installation documentation or see your MANTIS system administrator for more information.

If the DB2 BIND and GRANT are successful, DB2 returns an SQLCODE value of zero. You can verify that the Application Plan was created successfully by checking the DB2 Catalog for the Application Plan.

Making SQL support load modules available to MANTIS

The final step in preparing a program for static execution mode is making the SQL Support Load Module available to MANTIS. Making the SQL Support Load Module available to MANTIS may require system maintenance (CICS table maintenance, etc.).

SQL Support Load Modules can be associated with MANTIS in one of the following three ways:



Not all methods are available (or needed) in all environments. CICS MANTIS supports all three methods.

- ◆ **Included (linkedited) with MANTIS.** The SQL Support Load Module is linkedited with MANTIS and becomes part of the MANTIS load module. This method provides the best performance because MANTIS can directly access the SQL Support Load Module. The disadvantage of included modules is MANTIS load module size limits (512K maximum for some releases of CICS) and increased maintenance to the MANTIS load module.
- ◆ **Temporarily loaded.** The SQL Support Load Module is dynamically loaded when needed. For CICS MANTIS, the SQL Support Load Module is retained in memory only for the duration of the CICS task. For CICS pseudo-conversational tasks, the SQL Support Load Module is deleted at each terminal I/O or at program termination. For CICS conversational tasks, the SQL Support Load Module is deleted at program termination.
- ◆ **Permanently loaded.** The SQL Support Load Module is not linkedited with MANTIS. It is installed in the operating environment (CICS) as a program and dynamically loaded when required. For CICS, the module is loaded by specifying the CICS "HOLD" option, causing it to remain in memory, as long as CICS is running.

When choosing between included, temporarily loaded, or permanently loaded SQL Support Modules, consider the following factors:

- ◆ How often a static program will be used
- ◆ The size of SQL Support Load Module(s)
- ◆ The maximum load module size for your teleprocessing monitor
- ◆ Performance requirements versus ease of maintenance and likelihood of change

If you use an SQL Support Module infrequently, consider temporarily loading it. If you use it often, determine if including it with MANTIS will exceed your teleprocessing monitor's capacity (for example, 512K for some CICS releases) to help you decide whether to include it or permanently load it. Also, if your SQL Support Module changes frequently, temporarily or permanently loading it will provide easier maintenance.

Including load modules

To include an SQL Support Load Module with MANTIS, you must add the SQL Support Load Module to the INCLUDE list of the MANTIS CSOPCUST module and update the MANTIS linkdeck with an INCLUDE statement for the SQL Support Load Module.

The CSOPCUST module is created by the C\$OPCUST macro. This macro is explained in *MANTIS Administration, OS/390, VSE/ESA, P39-5005*. The SQL Support Load Module name must be placed in the C\$OPCUST INCLUDE parameter. C\$OPCUST must then be executed to create a new CSOPCUST module that must then be assembled.

You must manually update the linkdeck for MANTIS with an INCLUDE statement for the SQL Support Load Module. Sample linkdecks for MANTIS are supplied with the MANTIS installation tape. To modify the specific linkdeck, refer to the installation documentation or see your system administrator.

After you generate and assemble an updated CSOPCUST module and modify the linkdeck for MANTIS, linkedit MANTIS using the updated linkdeck to include the new CSOPCUST and SQL Support Load Modules.

For CICS MANTIS, be certain to linkedit the correct version of the SQL Support Load Module. The sample JCL supplied with MANTIS to precompile the SQL Support Load Module creates two SQL Support Load Modules: one for online (linkedited with the DB2 DSNCLI module) and one for batch (linkedited with the DB2 DSNELI module). Be sure that the online copy of the SQL Support Load Module is only linkedited with CICS MANTIS and that the batch copy of the SQL Support Load Module is only linkedited with Batch MANTIS. (Either copy of the SQL Support Load Module can be successfully linkedited with either CICS or Batch MANTIS. However, errors or abends will occur when the MANTIS program using the SQL Support Load Module is executed.)

Permanently loading SQL support load modules

To permanently load an SQL Support Load Module in CICS MANTIS, you must update the INCLUDE list in the CSOPCUST module, generate and assemble an updated CSOPCUST module, and re-linkedit MANTIS (see [“Including load modules”](#) on page 101). You do not need to modify the MANTIS linkdeck for permanently loaded SQL Support Load Modules.

The following CICS maintenance is required for permanently loaded SQL Support Load Modules:

- ◆ A PPT entry must be made for the SQL Support Load Module (a sample CICS PPT entry is distributed in the installation tape).
- ◆ The SQL Support Load Module must be placed in the CICS DFHRPL library.

To use permanently loaded SQL Support Load Modules in Batch MANTIS, enter the name of the library containing the SQL Support Load Module in the STEPLIB JCL statement for the Batch MANTIS job.

General considerations

- ◆ When linkediting MANTIS with the updated CSOPCUST module, the linkage editor will issue an IEW0461 error message for each permanently loaded SQL Support Load Module. You can ignore this message because the NCAL option was specified on the linkedit.
- ◆ For CICS MANTIS, be certain that the proper version of the SQL Support Load Module is available to CICS or Batch MANTIS. CICS and Batch MANTIS use different versions of the SQL Support Load Module (see the considerations for included SQL Support Load Modules in [“Permanently loading SQL support load modules”](#) on page 102). Make sure that the online version of the SQL Support Load Module is placed in the CICS DFHRPL library and that the STEPLIB DD statement (and any concatenation to it) in the Batch MANTIS job specifies the library containing the batch version of the SQL Support Load Module. (You can load either version of the SQL Support Load Module by CICS or Batch MANTIS. However, errors or abends will occur when you execute the MANTIS program using the SQL Support Load Module.)

Temporarily loading SQL support load modules

To temporarily load an SQL Support Load Module, you do not need to modify the MANTIS CSOPCUST module or re-linkedit MANTIS. System maintenance as described below is required.

For CICS MANTIS:

- ◆ A PPT entry must be made for the SQL Support Load Module (a sample CICS PPT entry is distributed on the MANTIS installation tape).
- ◆ The SQL Support Load Module must be placed in the CICS DFHRPL library.

For Batch MANTIS, enter the name of the library containing the SQL Support Load Module on the STEPLIB statement of the MANTIS job.

General consideration

For CICS and Batch MANTIS, be certain to place the proper version of the SQL Support Load Module in the DFHRPL or STEPLIB library (see considerations for permanently loaded SQL Support Load Modules in [“Permanently loading SQL support load modules”](#) on page 102).

Checking the consistency of a bound program

You can check the consistency of the program you have prepared to run in static execution mode to ensure that all preparation steps have been performed correctly and that all required entities are consistent with one another. This check compares the following information from the SQL bound MANTIS program and the SQL Support Load Module to ensure that they are the same:

- ◆ Date of MANTIS SQL Bind
- ◆ Time of MANTIS SQL Bind
- ◆ MANTIS user name
- ◆ MANTIS program name
- ◆ Number of SQL statements
- ◆ MANTIS release number
- ◆ Execution mode (static)
- ◆ Database type (DB2)

To perform a consistency check, select the SQL Check option from the Program Design Facility menu (see the screen illustration under “[SQL binding a MANTIS program online](#)” on page 83). Enter the name of the program at the SQLCHECK Program Entry panel as shown in the following screen illustration and press PF6 to execute.

```

PRGMENT101E      SQLCHECK Program Entry      YYYY/MM/DD HH:MM:SS
====>  _-
From
  Library . . . . TEST
  Name . . . .  SQL_DB2      Password :           :
  Description .

Thru
  Name . . . .

Entry Options      Function Options      Process Statistics

Immediate? . . . . Y      Display status? . . Y      Processed . .
Confirmation? . . . Y      Skipped . . .
Addendum? . . . . N      Errors . . . .

SQLBIND SQL_DB2

U01: CONFIRM OR SKIP
F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F6=EXECUTE F7=CONFIRM ...

```

MANTIS then returns the SQL Bind Information panel shown in the following screen illustration which displays either a confirmation that your program is consistent or the appropriate error message. See “[SQL binding a MANTIS program online](#)” on page 83 for field descriptions.

```
SQLBIND                SQL BIND INFORMATION PANEL

ACTION.....          : SQLBIND      :
MANTIS PROGRAM NAME.. : SQL_DB2          :
SQL BINDING TYPE..... : STATIC          :
SQL STATEMENTS.....  : 5              :

BOUND BY.....        : TEST            :
BOUND AT.....        : 92/09/22 09:15:37:
MANTIS RELEASE.....  :                :

SQL MODULE NAME ..... : DRVTST3        :
SQL MODULE OWNER NAME :                :
SQL OWNER PASSWORD    :                :
SQL DATA BASE TYPE .. : DB2            :

ENTER=CONTINUE  PF3=EXIT  PF6=EXECUTE  PA2=CANCEL
```

If the consistency check fails, MANTIS provides additional information on sources of error. You may need to retrace your path through the preparation steps to determine whether any steps were omitted or performed out of sequence. Check the SQL Support Load Modules to see if any were deleted, renamed, or modified. Make sure you have access to all the libraries that hold these entities. If you need to repeat a step, you must repeat all subsequent steps as well.

Unbinding SQL bound programs

You can unbind an SQL program and execute it in dynamic execution mode whenever you want. You may want to do so to compare the performance of dynamic execution mode with static execution mode, or if you want to modify and execute the program without preparing it for static execution mode.

You can unbind an SQL program in several ways:

Select the SQL Unbind option from the Program Design Facility menu (see the screen illustration under “[SQL binding a MANTIS program online](#)” on page 83). You will receive the SQLUNBIND Program Entry panel shown in the following screen illustration. Enter the name of the program you want to unbind and press PF6 to execute.

```

PRGMENT101E          SQLUNBIND Program Entry          YYYY/MM/DD HH:MM:SS
====>
From
  Library . . .      TEST
  Name . . . .      sql_db2
  Description .
                          Password :           :

Thru
  Name . . . .

Entry Options          Function Options          Process Statistics
Immediate? . . . . Y      Display status? . . Y      Processed . .
Confirmation? . . N
Addendum? . . . . N      Errors . . . .

000: READY
F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F6=EXECUTE F7=CONFIRM ...

```

The next panel displayed is the SQL Bind Information Panel, which displays status information about the program you are unbinding. Press PF6 to execute and MANTIS returns a message, NUCQBAI: SQL UNBIND COMPLETED SUCCESSFULLY, confirming the unbind as shown in the following screen illustration. See “SQL binding a MANTIS program online” on page 83 for a description of the fields on this panel.

```
SQLBIND                SQL BIND INFORMATION PANEL

ACTION..... : SQLBIND  :
MANTIS PROGRAM NAME.. : SQL_DB2    :
SQL BINDING TYPE..... : STATIC     :
SQL STATEMENTS.....  : 5          :

BOUND BY..... : TEST      :
BOUND AT..... : 92/09/22 09:15:37 :
MANTIS RELEASE..... :          :

SQL MODULE NAME ..... : DRVTST3   :
SQL MODULE OWNER NAME :           :
SQL OWNER PASSWORD   :           :
SQL DATA BASE TYPE .. : DB2      :
```

ENTER=CONTINUE PF3=EXIT PF6=EXECUTE PA2=CANCEL
NUCQBAI:SQL UNBIND COMPLETED SUCCESSFULLY

You can also unbind an SQL program by editing the program, changing it and replacing it.

General considerations

- ◆ If you unbind an SQL program and want to run it in static execution mode again you will have to repeat all of the preparation steps for static execution mode.
- ◆ If you change an SQL program that has been bound, MANTIS automatically unbinds it temporarily. If you RUN the modified program in programming mode, it will execute in dynamic execution mode rather than in static execution mode. If you REPLACE the modified program, it is permanently unbound. If you do not REPLACE the modified program, the program remains SQL bound.

SQL Bind Information continues to the right of the SQL Bind Information panel, beyond the MANTIS PROGRAM NAME field (see the preceding screen illustration). The additional information displayed includes the MANTIS release number, the number of SQL statements in the referenced program and the SQL database used by the program. To view this information enter Window Mode and use the appropriate PF keys to move around the panel. Press PF9 to exit Window Mode. (For information on Window Mode, refer to *MANTIS Language, OS/390, VSE/ESA*, P39-5002.)

To purge SQL Bind Information for a MANTIS program, enter P in the A (action) column (see the preceding screen illustration) and press ENTER. Confirm the purge by pressing PF4.

The Master User can view SQL Bind Information created by another user by entering the user's name in the <mantis-user> field at the upper left of the panel and pressing ENTER. The Master User may also purge another user's SQL Bind Information.

General considerations

- ◆ SQL Bind Information is required to generate SQL Support Source Modules.
- ◆ SQL Bind Information is automatically purged during SQL Support Source Module generation if DELETE is set to YES. This parameter may be set either explicitly or by default (see “*Generating the SQL support source module*” on page 93).

Running SQL support programs with batch MANTIS

Programs that access DB2 must use one of three attach-facility programs provided by IBM. These attach-facility programs exist for CICS (online), IMS (online and batch) and TSO (batch). For CICS users to execute MANTIS SQL Support Programs in batch, the TSO Terminal Monitor Program (TMP) must be used to invoke MANTIS so that the TSO DB2 attach-facility program can be used. This requires some changes to the usual OS/390 JCL needed to execute batch MANTIS. For IMS users, batch MANTIS can execute using the IMS BMP, DLI or DBB facilities.

Sample Batch MANTIS JCL for both TSO and IMS are included on the MANTIS installation tape. For more information, refer to the installation documentation or see your system administrator.

For TSO, be sure that the STEPLIB DD statement (and any concatenation to it) specifies the proper library containing the batch version of the SQL Support Load Module (see [“Temporarily loading SQL support load modules”](#) on page 103).

6

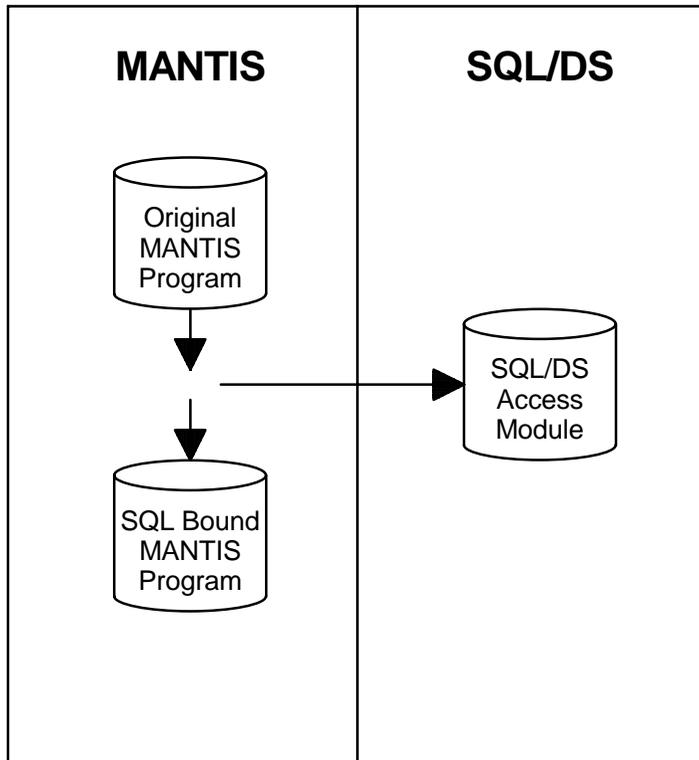
MANTIS SQL support programs for extended dynamic execution mode (DB2 for VSE and VM—formerly SQL/DS)

Extended dynamic execution mode allows the SQL statements in MANTIS SQL programs to be compiled into a DB2 for VSE and VM Access Module just as they are for SQL in COBOL. The procedure MANTIS uses to create the Access Module is different than that used by SQL in COBOL. SQL in COBOL requires each program to be precompiled and compiled before being executed. The DB2 for VSE and VM Access Module is created by the precompiler program during the precompile step. However, MANTIS SQL Support uses extended dynamic SQL statements to create the Access Module. Extended dynamic statements allow MANTIS to perform the functions that are normally done by the precompiler program. Consequently, no precompile or compile is necessary for MANTIS SQL Support programs to execute in extended dynamic execution mode. (For more information on extended dynamic SQL statements, refer to the DB2 for VSE and VM documentation.)

In dynamic execution mode, MANTIS executes all SQL statements by dynamic SQL statements, eliminating the SQL precompile step each time the program is executed. You can code SQL statements in a MANTIS program and immediately RUN the program. When errors are detected, you can modify these statements and immediately re-execute the program. The flexibility of dynamic execution mode requires considerable computer resources. While using these resources is desirable during program development, they should be limited when the program is ready for production. In MANTIS SQL Support for DB2 for VSE and VM, executing the program in extended dynamic execution mode instead of dynamic execution mode uses fewer resources.

Before you can execute a MANTIS SQL program in extended dynamic execution mode, you must SQL bind it. (Do not confuse SQL binding with HPO (High Performance Option) binding. Both HPO bound and unbound programs can be SQL bound.) MANTIS SQL programs are SQL bound using the SQL Bind Facility (see “[SQL binding a MANTIS program online](#)” on page 118). During the SQL bind, MANTIS SQL Support uses extended dynamic SQL statements to create the DB2 for VSE and VM Access Module. See the following figure for an illustration of this process. When the bind is complete, an Access Module containing all the SQL statements in the MANTIS program has been created, and the MANTIS program has been marked as SQL bound. The program is ready for immediate execution in extended dynamic execution mode.

The following figure shows the step required to run a MANTIS program in extended dynamic execution mode:



Performance and programming considerations

Programs executing in extended dynamic execution mode cannot always execute exactly as they do in dynamic execution mode. This section lists certain differences and restrictions for programs executing in extended dynamic execution mode. Review these differences before preparing your program for execution in extended dynamic mode.

Determining host variable data types

When a MANTIS program is SQL bound, MANTIS temporarily HPO binds the program (if it is not already HPO bound). This binding determines the data type and length of MANTIS variables used as host variables in SQL statements. You must know this information so that you can perform the precompile and assembly. MANTIS considers any host variables that are not resolved by the HPO bind to be undefined and may default to BIG, depending on the option you specify when you execute the SQL Bind option (see [“SQL binding a MANTIS program online”](#) on page 118).

Because HPO binding determines host variable data types, structure the MANTIS program as if it were to be HPO bound. Define all MANTIS variables that are used as host variables in the program before any statements that cause HPO binding to terminate. This is especially true for nonnumeric variables (TEXT and KANJI). For more information on HPO binding, refer to [MANTIS Facilities, OS/390, VSE/ESA, P39-5001](#). If a MANTIS TEXT variable defaults to BIG in the DB2 for VSE and VM Access Module, the program will not run in extended dynamic execution mode. Either the SQL database or MANTIS will return an error, depending on how the text variable is used.

Multiple row result sets with SELECT

In MANTIS dynamic and extended dynamic execution modes, a SELECT statement always returns the first row of the result set, regardless of the number of rows in the result set. In the example below, the first row in the result set built by the SELECT is always returned, even if the result set contains more than one row.

```
EXEC_SQL  
. | SELECT * INTO :X, :Y FROM SQLTABLE  
. | WHERE SQLCOLUMN > :Z  
END
```

You must use an SQL CURSOR statement if you want to retrieve all of the rows built by the SELECT in the result set. The following example is the equivalent of the previous example, using SQL CURSORS instead of SELECT.

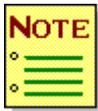
```
EXEC_SQL  
. | DECLARE C1 CURSOR FOR SELECT *  
. | FROM SQLTABLE WHERE SQLCOLUMN > :Z  
END  
. |  
EXEC_SQL  
. | OPEN C1  
END  
. |  
EXEC_SQL  
. | FETCH C1 INTO :X, :Y  
END  
. |  
EXEC_SQL  
. | CLOSE C1  
END
```

Binding on the target system

SQL bind the MANTIS program on the system where it will be executed or a clone of that system because SQL binding uses HPO binding to determine host variable data types. If MANTIS variables that are used as host variables are defined in HPO bound entities (screens, files, etc.), and these entities are different on the target system from those on the system where the SQL bind is done, unpredictable results, errors, or abends may occur.

Preparing a program to run in extended dynamic execution mode

To prepare a MANTIS program to run in extended dynamic execution mode, you must SQL bind the program using the MANTIS SQL Bind facility. Bind Options for SQL programs are accessed from the MANTIS Program Design Facility menu shown in the screen illustration under “[SQL binding a MANTIS program online](#)” on page 118. The options include binding, unbinding, and checking maintenance actions. If you are unfamiliar with the Program Design Facility or the Bind Options, refer to [MANTIS Program Design and Editing, OS/390, VSE/ESA](#), P39-5013, for details.



The SQL Maint option is not functional for programs using DB2 for VSE and VM or SQL/DS. If you select this option, you will receive the error message NUCQBUE: SQL DATA BASE DOES NOT SUPPORT FUNCTION - SQL MAINT. (For information on error messages, refer to [MANTIS Messages and Codes, OS/390, VSE/ESA](#), P39-5004.)

SQL binding a MANTIS program online

To SQL bind a MANTIS program, select the SQL Bind option at the Program Design Facility menu as shown in the following screen illustration and press ENTER:

```
PRGMMENU01      Program Design Facility (TEST)      YYYY/MM/DD HH:MM:SS
====>

Please select one of the menu items below.

      Program      Component Engineering  Bind Options  Utilities
16  1. List        7. CEF Check      12. HPO Check 18. Audit Trail
    2. Edit        8. " Compose      13. " Bind    19. Browse Audit Trail
    3. Profile     9. " Decompose    14. " Unbind  20. " Prgm Profile
    4. Purge      10. CREF Programs 15. SQL Check 21. Trigger List
    5. Copy       11. Bill of Materials 16. " Bind    22. SQL Maint
    6. Rename     17. " Unbind

000: READY
F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F9=RETRIEVE F12=CANCEL ...
```

The next panel displayed is the SQLBIND Program Entry panel (see the following screen illustration). This panel allows you to specify the program or programs you want to SQL bind. You can specify a single program name, a range of program names, or a generic pattern of names. You can also specify Entry Options, Function Options and Process Statistics.

The Function Options for the SQLBIND Program Entry panel are described below. For information on the Entry Options or Process Statistics, refer to *MANTIS Facilities, OS/390, VSE/ESA*, P39-5001. In this example, a single program name is specified and the defaults for all other options are accepted. Press PF6 to execute.

```

PRGMENT101E      SQLBIND Program Entry                      YYYY/MM/DD HH:MM:SS
===>
From
  Library . . . . TEST
  Name . . . . SQL_PRGMI          Password :           :
  Description .

Thru
  Name . . . .

Entry Options          Function Options          Process Statistics
Immediate? . . . . Y   Preprocess Options:      Processed . .
Confirmation? . . . . Y   . .Save/Replace . .     S Skipped . . .
Addendum? . . . . N     . .Keep/Revoke . .     K Errors . . . .
                        . .Block/Noblock . .     N
                        Undefined Variables?     1

000: READY
F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F6=EXECUTE F7=CONFIRM ...
    
```

PREPROCESS OPTIONS:

Description *Required.* Specifies the options to be used for the SQL bind.

SAVE/REPLACE

Description *Required.* Specifies whether the Access Module created during the SQL bind is to be added to the DB2 for VSE and VM catalog ("S"), or is to replace an existing Access Module ("R").

Default SAVE ("S")

Consideration An error will be returned if "S" is specified for an Access Module that already exists.

KEEP/REVOKE

Description *Required.* Indicates whether the RUN authorities granted to the existing Access Module should be kept ("K") for the new Access Module or revoked ("R").

Default KEEP ("K")

BLOCK/NOBLOCK

Description *Required.* Specifies whether rows retrieved by the Access Module will be grouped together in blocks ("B") to improve performance, or accessed individually ("N").

Default NOBLOCK ("N")

Consideration Some SQL statements cannot be executed when BLOCK is specified. See your DB2 for VSE and VM documentation for more information.

UNDEFINED VARIABLES?

Description	<i>Required.</i> Specifies what should be done if undefined variables are encountered during the SQL Bind.
Default	1
Options	<ol style="list-style-type: none">1 QUIT WITH DISPLAY. If any undefined variables are found, MANTIS displays the undefined variables with their line numbers and does not SQL bind the program. These variables must be changed in the MANTIS program or another option must be selected. If no undefined variables are found, the program is SQL bound.2 CONTINUE NO DISPLAY. Binding continues even if undefined variables are encountered. All undefined variables are assigned a data type of BIG, but these variables are not displayed.3 CONTINUE WITH DISPLAY. Binding continues even if MANTIS encounters undefined variables. Undefined variables are assigned a data type of BIG and displayed with their line numbers.

Considerations

- ◆ Nonnumeric variables that default to BIG can create inconsistencies that prevent extended dynamic mode execution (see ["Binding on the target system"](#) on page 117).
- ◆ Any undefined variables displayed include the line number on which they appear. Undefined variables are not displayed with option 2.

When you press PF6 to execute, MANTIS returns the description of the program you have specified for binding and asks you to either CONFIRM or SKIP the action as shown in the following screen illustration. Press PF7 to confirm the action.

```
PRGMENT101E      SQLBIND Program Entry      YYYY/MM/DD HH:MM:SS
====>
From
  Library . . . . TEST
  Name . . . . . SQL_PRGM1      Password :
  Description . . DYNAMIC SQL/DS ACCESS PROGRAM

Thru
  Name . . . . .

Entry Options      Function Options      Process Statistics
Immediate? . . . . Y      Preprocess Options:      Processed . .
Confirmation? . . Y      . .Save/Replace . .      S Skipped . . .
Addendum? . . . . N      . .Keep/Revoke . .      K Errors . . . .
                          . .Block/Noblock . .      B
                          Undefined Variables?      1

SQLBIND      SQL_PRGM1

U01: CONFIRM OR SKIP
F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F6=EXECUTE F7=CONFIRM . . .
```

The next panel displayed is the SQL Bind Information panel (see the following screen illustration). The options for this panel are described below. After making any changes or specifications necessary, press PF6 to complete the binding process. If no errors are encountered, a confirmation message, NUCQBAI: SQL BIND COMPLETED SUCCESSFULLY, will be returned. Press PA2 to return to the Program Design Facility menu. Press PF3 to exit and return to the MANTIS Facility Selection menu.

```

SQLBIND                SQL BIND INFORMATION PANEL

ACTION.....          : SQLBIND      :
MANTIS PROGRAM NAME.. : SQL PRGM1                          :
SQL BINDING TYPE..... : EXTENDED DYNAMIC :
SQL STATEMENTS.....  : 5                :

BOUND BY.....        : TEST                :
BOUND AT.....        : 91/09/22 09:15:37 :
MANTIS RELEASE.....  :                    :

SQL MODULE NAME ..... : DRVTST3           :
SQL MODULE OWNER NAME :                   :
SQL OWNER PASSWORD    :                   :
SQL DATA BASE TYPE .. : SQL/DS           :

ENTER=CONTINUE  PF3=EXIT  PF6=EXECUTE  PA2=CANCEL
    
```

MANTIS PROGRAM NAME

Description *Protected.* Displays your user name and the name of the MANTIS SQL program you are binding.

SQL BINDING TYPE

Description *Protected.* Displays the type of SQL bind that will be performed (extended dynamic).

SQL STATEMENTS

Description *Protected.* Displays the number of SQL statements in the program you are binding.

SQL MODULE NAME

- Description** *Required.* The name of the DB2 for VSE and VM Access Module. If the MANTIS SQL program was previously bound, the name of the previously created DB2 for VSE and VM Access Module is displayed; otherwise, this field is blank.
- Default** The existing Access Module name (if previously bound).
- Format** 1-8 alphanumeric character symbolic name. The first character may be A-Z, @, #, \$; characters 2-7 may be these characters plus 0-9.
- Consideration** The name must be unique to the SQL database and to MANTIS.

SQL MODULE OWNER NAME

- Description** *Optional.* Indicates the DB2 for VSE and VM owner of the Access Module.
- Default** Current MANTIS user (if unbound) or previous owner (if bound).
- Consideration** If this field is input, MANTIS will execute an SQL CONNECT to this DB2 for VSE and VM user before the SQL bind begins, specifying SQL OWNER PASSWORD as the SQL password. When the SQL bind is complete, MANTIS will execute a COMMIT WORK RELEASE.

SQL OWNER PASSWORD

- Description** *Optional.* Specifies a valid DB2 for VSE and VM user password.
- Default** None.
- Consideration** This field may be required if the SQL MODULE OWNER NAME field is input. See the consideration for SQL MODULE OWNER NAME above.

SQL DATA BASE TYPE

- Description** *Protected.* The database for which the bind will be done.
- Default** SQL/DS
- Consideration** DB2 for VSE and VM (formerly SQL/DS) is currently the only database that supports extended dynamic execution mode.

SQL binding a MANTIS program with batch MANTIS

You can SQL bind one or more SQL programs by using Batch MANTIS. Sample input for SQL binding with Batch MANTIS is in the following screen illustration. Sample JCL to SQL bind MANTIS programs using Batch MANTIS is supplied with the MANTIS installation tape. Refer to the installation documentation or contact your system administrator for more information. Note that the asterisk (*) denotes three separate fields.

```
<ECHO=ON>user;password;
<FAULT=OFF>;
<BLANK=OFF>;1;
CONTROL:SQLBIND_FACILITY;

user:program1;name1;owner;preprocess_option*;when_undefined_option;password;

user:program2;name2;owner;preprocess_option*;when_undefined_option;password;
.
.
<PA2>
<PA2>
```

Field equivalents

name 1	SQL_module_name1
owner	SQL_module_owner
preprocess_option	preprocessing_option (3 fields)
password1	SQL_owner_password1
name 2	SQL_module_name2
password2	SQL_owner_password2

General considerations

- ◆ Because the MANTIS program is updated during an SQL bind, the MANTIS cluster must be updateable by Batch MANTIS.
- ◆ The output is routed to the printer, so you may want to place the PRINTER DD card ahead of the TERMINAL DD card or use TERMINAL DD DUMMY.
- ◆ If you process multiple programs that contain undefined variables in batch, the undefined variables are printed on the page before the panel containing their program name.

Checking the results of an SQL bind

To see whether an SQL program is bound, select option 1 (Program List) from the Program Design Facility menu (see the screen illustration under “SQL binding a MANTIS program online” on page 118) and press ENTER. The next panel displayed is the Program Directory List shown in the following screen illustration. If the program has been bound, the BND field will contain an E, meaning that the program is bound for extended dynamic execution.

For more information on the Program Directory List and a description of the fields, refer to *MANTIS Program Design and Editing, OS/390, VSE/ESA*, P39-5013.

```

PRGMLIST01          Program Directory List                      YYYY/MM/DD HH:MM:SS
====>
From _____
Action   Name                               Date      Time      Ver BND Status
-----
_____ SQL_PRGM                          92/02/22 09:15:37  5      ACTIVE
_____ SQL_PRGM1                         92/02/22 09:15:37  5      E ACTIVE
_____ SQL_PRGM2                         92/02/22 09:15:37  5      ACTIVE
_____ SQL_PRGM3                         92/02/22 09:15:37  5      ACTIVE
_____ SQL_PRGM4                         92/02/22 09:15:37  5      ACTIVE
_____ SQL_PRGM5                         92/02/22 09:15:37  5      ACTIVE
.
.
.

F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F8=FWD F9=RETRIEVE...
```

Checking the consistency of a bound program

You can check the consistency of the program you have prepared to run in extended dynamic execution mode to ensure that all preparation steps have been performed correctly and that all required entities are consistent with one another. This check compares the following information from the SQL bound MANTIS program and DB2 for VSE and VM (or SQL/DS) Access Module to ensure that they are the same:

- ◆ Date of MANTIS SQL Bind
- ◆ Time of MANTIS SQL Bind
- ◆ MANTIS user name
- ◆ MANTIS program name
- ◆ Number of SQL statements
- ◆ MANTIS release number
- ◆ Execution mode (extended dynamic)
- ◆ Database type (SQL/DS)

To perform a consistency check, select the SQL Check option from the Program Design Facility menu (see the screen illustration under “[SQL binding a MANTIS program online](#)” on page 118). Enter the name of the program at the SQLCHECK Program Entry panel as shown in the following screen illustration and press PF6 to execute.

```

PRGMENT101E      SQLCHECK Program Entry                      YYYY/MM/DD HH:MM:SS
====>  _
From
  Library . . . . TEST
  Name . . . . . SQL_PRGM1                      Password :           :
  Description .

Thru
  Name . . . . .

Entry Options          Function Options          Process Statistics

  Immediate? . . . . Y          Display status? . . . Y    Processed . .
  Confirmation? . . . Y          Skipped . . .
  Addendum? . . . . N          Errors . . . .

SQLBIND      SQL_DB2

U01: CONFIRM OR SKIP
F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F6=EXECUTE F7=CONFIRM ...

```

MANTIS then returns the SQL Bind Information panel shown in the following screen illustration which displays either a confirmation that your program is consistent or the appropriate error message. See “[SQL binding a MANTIS program online](#)” on page 118 for field descriptions.

```
SQLBIND                SQL BIND INFORMATION PANEL

ACTION.....          : SQLCHECK  :
MANTIS PROGRAM NAME.. : SQL PRGM1                      :
SQL BINDING TYPE..... : EXTENDED DYNAMIC :
SQL STATEMENTS.....  : 5          :

BOUND BY.....        : TEST          :
BOUND AT.....        : 91/09/22 09:15:37:
MANTIS RELEASE.....  :              :

SQL MODULE NAME ..... : DRVTST3      :
SQL MODULE OWNER NAME :              :
SQL OWNER PASSWORD   :              :
SQL DATA BASE TYPE .. : SQL/DS      :

ENTER=CONTINUE  PF3=EXIT  PF6=EXECUTE  PA2=CANCEL
```

If the consistency check fails, MANTIS provides additional information on sources of error. You may need to check the DB2 for VSE and VM Access Module to see if it was deleted, renamed, or modified.

Unbinding SQL bound programs

You can unbind an SQL program and execute it in dynamic execution mode whenever you want. You may want to do so to compare the performance of dynamic execution mode with extended dynamic execution mode, or if you want to modify and execute the program without preparing it for extended dynamic execution mode.

You can unbind an SQL program in several ways:

Select the SQL Unbind option from the Program Design Facility menu (see the screen illustration under “[SQL binding a MANTIS program online](#)” on page 118). You will receive the SQLUNBIND Program Entry panel shown in the following screen illustration. Enter the name of the program you want to unbind and press PF6 to execute.

```

PRGMENT101E          SQLUNBIND Program Entry          YYYY/MM/DD HH:MM:SS
====>
From
  Library . . .      TEST
  Name . . . .      SQL_PRGM1          Password :          :
  Description .

Thru
  Name . . . .

Entry Options          Function Options          Process Statistics
Immediate? . . . . Y      Display status? . . Y      Processed . .
Confirmation? . . N          Skipped . . .
Addendum? . . . . N          Errors . . . .

000: READY
F1=HELP F2=EXHELP F3=EXIT F4=PROMPT F5=REFRESH F6=EXECUTE F7=CONFIRM ...

```

The next panel displayed is the SQL Bind Information Panel, which displays status information about the program you are unbinding. If the DB2 for VSE and VM user shown in the SQL MODULE OWNER NAME requires a password to access resources owned by this user, enter the password in SQL OWNER PASSWORD. During the unbind, MANTIS SQL Support will execute an SQL CONNECT to the DB2 for VSE and VM user contained in the SQL MODULE OWNER NAME field, specifying the password contained in the SQL OWNER PASSWORD field. When the unbind is complete, a COMMIT WORK RELEASE will be executed.

Press PF6 to execute and MANTIS returns a message, NUCQBAI: SQL UNBIND COMPLETED SUCCESSFULLY, confirming the unbind as shown in the following screen illustration. See “SQL binding a MANTIS program online” on page 118 for a description of the fields on this panel.

```
SQLBIND                SQL BIND INFORMATION PANEL

ACTION.....          : SQLBIND      :
MANTIS PROGRAM NAME.. : SQL PRGM1      :
SQL BINDING TYPE....  : EXTENDED DYNAMIC :
SQL STATEMENTS.....  : 5              :

BOUND BY.....        : TEST           :
BOUND AT.....        : 91/09/22 09:15:37 :
MANTIS RELEASE.....  :                :

SQL MODULE NAME .... : DRVTST3       :
SQL MODULE OWNER NAME : SQLDBA        :
SQL OWNER PASSWORD   :               :
SQL DATA BASE TYPE .. : SQL/DS        :

ENTER=CONTINUE PF3=EXIT PF6=EXECUTE PA2=CANCEL
NUCQBAI:SQL UNBIND COMPLETED SUCCESSFULLY
```

You can also unbind an SQL program by editing the program, changing it and replacing it.

General considerations

- ◆ If you unbind an SQL program and want to run it in extended dynamic execution mode again you must SQL bind the program again.
- ◆ If you change an SQL program that has been bound, MANTIS automatically unbinds it temporarily. If you RUN the modified program in programming mode, it will execute in dynamic execution mode rather than in extended dynamic execution mode. If you REPLACE the modified program, it is permanently unbound. If you do not REPLACE the modified program, the program remains SQL bound.
- ◆ If a bound program is modified and replaced, it is permanently unbound (see above). The DB2 for VSE and VM Access Module for the program may be deleted. If the program is SQL bound again, the SAVE/REPLACE options of the SQLBIND panel may require “R” instead of “S” (see [“SQL binding a MANTIS program online”](#) on page 118).

A

Sample MANTIS SQL support programs

This appendix contains the following sample MANTIS SQL Support programs:

- ◆ Insert program using static SQL statements
- ◆ Insert program using dynamic SQL statements
- ◆ Update program using static SQL statements
- ◆ Update program using dynamic SQL statements
- ◆ Select program using static SQL statements
- ◆ Select program using dynamic SQL statements
- ◆ Delete program using static SQL statements
- ◆ Delete program using dynamic SQL statements
- ◆ Column select program using dynamic SQL statements
- ◆ Holding cursors over a COMMIT statement

Each example program using dynamic SQL statements is equivalent to the corresponding program using static SQL statements. For clarity, the examples do not contain error checking or display logic and the information to be transferred to the database is coded in the programs as literals.

The different SQL statement types (static and dynamic) should not be confused with MANTIS SQL Support execution modes (static, dynamic or extended dynamic). All example programs can be executed in any MANTIS SQL Support execution mode.

Insert program using static SQL statements

This program inserts one employee into an employee table using static SQL statements.

```
10 ENTRY STATIC_INSERT
20 .BIG HIRE_DATE,BIRTH_DATE,JOB_CODE,SALARY,EDUCATION_LEVEL
30 .TEXT EMPLOYEE_NUMBER(6),FIRST_NAME(20),MIDDLE_INITIAL(1),LAST_NAME(20)
40 .TEXT PHONE_NUMBER(4),WORK_DEPARTMENT(3),SEX(1)
50 .|
60 .EMPLOYEE_NUMBER="000120"
70 .FIRST_NAME="SEAN"
80 .MIDDLE_INITIAL=" "
90 .LAST_NAME="O'CONNELL"
100 .BIRTH_DATE=421018
110 .HIRE_DATE=631205
120 .JOB_CODE=58
130 .EDUCATION_LEVEL=14
140 .SALARY=29250
150 .PHONE_NUMBER="2167"
160 .WORK_DEPARTMENT="A00"
170 .SEX="M"
180 .|
```

```
190 .EXEC_SQL: | INSERT INTO DSN82.TEMPL
200 .. |           (EMPNO,
210 .. |           FIRSTNME,
220 .. |           MIDINIT,
230 .. |           LASTNAME,
240 .. |           BRTHDATE,
250 .. |           HIREDATE,
260 .. |           JOBCODE,
270 .. |           EDUCLVL,
280 .. |           SALARY,
290 .. |           PHONENO,
300 .. |           WORKDEPT,
310 .. |           SEX)
320 .. |           VALUES (:EMPLOYEE_NUMBER,
330 .. |           :FIRST_NAME,
340 .. |           :MIDDLE_INITIAL,
350 .. |           :LAST_NAME,
360 .. |           :BIRTH_DATE,
370 .. |           :HIRE_DATE,
380 .. |           :JOB_CODE,
390 .. |           :EDUCATION_LEVEL,
400 .. |           :SALARY,
410 .. |           :PHONE_NUMBER,
420 .. |           :WORK_DEPARTMENT,
430 .. |           :SEX)
440 .END
450 EXIT
```

Insert program using dynamic SQL statements

This program inserts one employee into an employee table using dynamic SQL statements.

```
10 ENTRY DYNAMIC_INSERT
20 .BIG HIRE_DATE,BIRTH_DATE,JOB_CODE,SALARY,EDUCATION_LEVEL
30 .TEXT EMPLOYEE_NUMBER(6),FIRST_NAME(20),MIDDLE_INITIAL(1),LAST_NAME(20)
40 .TEXT PHONE_NUMBER(4),WORK_DEPARTMENT(3),SEX(1)
50 .TEXT SQL_TEXT(254)
60 .|
70 .EMPLOYEE_NUMBER="000120"
80 .FIRST_NAME="SEAN"
90 .MIDDLE_INITIAL=" "
100 .LAST_NAME="O'CONNELL"
110 .BIRTH_DATE=421018
120 .HIRE_DATE=631205
130 .JOB_CODE=58
140 .EDUCATION_LEVEL=14
150 .SALARY=29250
160 .PHONE_NUMBER="2167"
170 .WORK_DEPARTMENT="A00"
180 .SEX="M"
190 .|
200 .SQL_TEXT="INSERT INTO DSN82.TEMPL "
210 .'"(EMPNO, FIRSTNME, MIDINIT, LASTNAME, BRTHDATE, "
220 .'"HIREDATE, JOBCODE, EDUCLVL, SALARY, PHONENO, "
230 .'"WORKDEPT, SEX) "
240 .'"VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"
250 .|
260 .EXEC_SQL:| PREPARE S1 FROM :SQL_TEXT
270 .END
280 .|
```

```
290 .SQLDA( "SQLDA1" )=NEW
300 .SQLDA( "SQLDA1" , "SQLN" )=12
310 .SQLDA( "SQLDA1" , "SQLD" )=12
320 .SQLDA( "SQLDA1" , "SQLDATA" , 1 )=EMPLOYEE_NUMBER
330 .SQLDA( "SQLDA1" , "SQLDATA" , 2 )=FIRST_NAME
340 .SQLDA( "SQLDA1" , "SQLDATA" , 3 )=MIDDLE_INITIAL
350 .SQLDA( "SQLDA1" , "SQLDATA" , 4 )=LAST_NAME
360 .SQLDA( "SQLDA1" , "SQLDATA" , 5 )=BIRTH_DATE
370 .SQLDA( "SQLDA1" , "SQLDATA" , 6 )=HIRE_DATE
380 .SQLDA( "SQLDA1" , "SQLDATA" , 7 )=JOB_CODE
390 .SQLDA( "SQLDA1" , "SQLDATA" , 8 )=EDUCATION_LEVEL
400 .SQLDA( "SQLDA1" , "SQLDATA" , 9 )=SALARY
410 .SQLDA( "SQLDA1" , "SQLDATA" , 10 )=PHONE_NUMBER
420 .SQLDA( "SQLDA1" , "SQLDATA" , 11 )=WORK_DEPARTMENT
430 .SQLDA( "SQLDA1" , "SQLDATA" , 12 )=SEX
440 .|
450 .EXEC_SQL:| EXECUTE S1 USING DESCRIPTOR SQLDA1
460 .END
470 EXIT
```

Update program using static SQL statements

This program updates one employee in an employee table using static SQL statements.

```
10 ENTRY STATIC_UPDATE
20 .BIG HIRE_DATE,BIRTH_DATE
30 .TEXT EMPLOYEE_NUMBER(6)
40 .TEXT FIRST_NAME(20),MIDDLE_INITIAL(1),LAST_NAME(20)
50 .|
60 .EMPLOYEE_NUMBER="000120"
70 .FIRST_NAME="JOHN"
80 .MIDDLE_INITIAL="H"
90 .LAST_NAME="DOE"
100 .BIRTH_DATE=490113
110 .HIRE_DATE=880120
120 .|
130 .EXEC_SQL
140 ..|
150 ..| UPDATE DSN82.TEMPL
160 ..|
170 ..| SET FIRSTNME = :FIRST_NAME,
180 ..|     MIDINIT   = :MIDDLE_INITIAL,
190 ..|     LASTNAME  = :LAST_NAME,
200 ..|     BRTHDATE  = :BIRTH_DATE,
210 ..|     HIREDATE  = :HIRE_DATE
220 ..|
230 ..| WHERE EMPNO = :EMPLOYEE_NUMBER
240 .END
250 EXIT
```

Update program using dynamic SQL statements

This program updates one employee in an employee table using dynamic SQL statements.

```

10 ENTRY DYNAMIC_UPDATE
20 .BIG HIRE_DATE,BIRTH_DATE
30 .TEXT EMPLOYEE_NUMBER(6),FIRST_NAME(20),MIDDLE_INITIAL(1),LAST_NAME(20)
40 .TEXT DA(18),DAPARM(8)
50 .TEXT SQL_TEXT(254)
60 .|
70 .EMPLOYEE_NUMBER="000120"
80 .FIRST_NAME="JOHN"
90 .MIDDLE_INITIAL="H"
100 .LAST_NAME="DOE"
110 .BIRTH_DATE=490113
120 .HIRE_DATE=880120
130 .|
140 .SQL_TEXT="UPDATE DSN82.TEMPL SET "
150 .SQL_TEXT=SQL_TEXT+"FIRSTNME = ?, MIDINIT = ?, LASTNAME = ?, "
160 .SQL_TEXT=SQL_TEXT+"BRTHDATE = ?, HIREDATE = ? "
170 .SQL_TEXT=SQL_TEXT+"WHERE EMPNO = ? "
180 .|
190 .EXEC_SQL:| PREPARE S1 FROM :SQL_TEXT
200 .END
210 .|
220 .SQLDA("SQLDA1")=NEW
230 .DA="SQLDA1"
240 .DAPARM="SQLDATA"
250 .SQLDA(DA,"SQLN")=6
260 .SQLDA(DA,"SQLD")=6
270 .SQLDA(DA,DAPARM,1)=FIRST_NAME
280 .SQLDA(DA,DAPARM,2)=MIDDLE_INITIAL
290 .SQLDA(DA,DAPARM,3)=LAST_NAME
300 .SQLDA(DA,DAPARM,4)=BIRTH_DATE
310 .SQLDA(DA,DAPARM,5)=HIRE_DATE
320 .SQLDA(DA,DAPARM,6)=EMPLOYEE_NUMBER
330 .|
340 .EXEC_SQL:| EXECUTE S1 USING DESCRIPTOR SQLDA1
350 .END
360 EXIT

```

Select program using static SQL statements

This program retrieves employee information for one employee from an employee table using static SQL statements.

```
10 ENTRY STATIC_SELECT
20 .BIG HIRE_DATE,BIRTH_DATE,JOB_CODE,SALARY,EDUCATION_LEVEL
30 .TEXT EMPLOYEE_NUMBER(6),FIRST_NAME(20),MIDDLE_INITIAL(1),LAST_NAME(20)
40 .TEXT WORK_DEPARTMENT(3),PHONE_NUMBER(3),SEX(1)
50 .EMPLOYEE_NUMBER="000120"
60 .|
70 .EXEC_SQL:| DECLARE C1 CURSOR FOR
80 ..|           SELECT * FROM DSN82.TEMPL
90 ..|           WHERE EMPNO = :EMPLOYEE_NUMBER
100 .END
110 .EXEC_SQL:| OPEN C1
120 .END
130 .EXEC_SQL:| FETCH C1 INTO  :EMPLOYEE_NUMBER,
140 ..|                       :FIRST_NAME,
150 ..|                       :MIDDLE_INITIAL,
160 ..|                       :LAST_NAME,
170 ..|                       :BIRTH_DATE,
180 ..|                       :HIRE_DATE,
190 ..|                       :JOB_CODE,
200 ..|                       :EDUCATION_LEVEL,
210 ..|                       :SALARY,
220 ..|                       :PHONE_NUMBER,
230 ..|                       :WORK_DEPARTMENT,
240 ..|                       :SEX
250 .END
260 .EXEC_SQL:| CLOSE C1
270 .END
280 EXIT
```

Select program using dynamic SQL statements

This program retrieves employee information for one employee from an employee table using dynamic SQL statements.

```

10 ENTRY DYNAMIC_SELECT
20 .BIG HIRE_DATE , BIRTH_DATE , JOB_CODE , SALARY , EDUCATION_LEVEL
30 .TEXT EMPLOYEE_NUMBER(6)
40 .TEXT FIRST_NAME(20) , MIDDLE_INITIAL(1) , LAST_NAME(20)
50 .TEXT WORK_DEPARTMENT(3) , PHONE_NUMBER(3) , SEX(1)
60 .TEXT SQL_TEXT(254)
70 .|
80 .EMPLOYEE_NUMBER="000120"
90 .SQL_TEXT="SELECT * FROM DSN82.TEMPL"
100 ." WHERE EMPNO = ? "
110 .SQLDA("SQLDA1")=NEW
120 .|
130 .EXEC_SQL:| PREPARE S1 FROM :SQL_TEXT
140 .END
150 .EXEC_SQL:| DESCRIBE S1 INTO SQLDA1
160 .END
170 .EXEC_SQL:| DECLARE C1 CURSOR FOR S1
180 .END
190 .EXEC_SQL:| OPEN C1 USING :EMPLOYEE_NUMBER
200 .END
210 .EXEC_SQL:| FETCH C1 USING DESCRIPTOR SQLDA1
220 .END
230 .EXEC_SQL:| CLOSE C1
240 .END
250 .|
260 .EMPLOYEE_NUMBER=SQLDA("SQLDA1", "SQLDATA", 1)
270 .FIRST_NAME=SQLDA("SQLDA1", "SQLDATA", 2)
280 .MIDDLE_INITIAL=SQLDA("SQLDA1", "SQLDATA", 3)
290 .LAST_NAME=SQLDA("SQLDA1", "SQLDATA", 4)
300 .BIRTH_DATE=SQLDA("SQLDA1", "SQLDATA", 5)
310 .HIRE_DATE=SQLDA("SQLDA1", "SQLDATA", 6)
320 .JOB_CODE=SQLDA("SQLDA1", "SQLDATA", 7)
330 .EDUCATION_LEVEL=SQLDA("SQLDA1", "SQLDATA", 8)
340 .SALARY=SQLDA("SQLDA1", "SQLDATA", 9)
350 .PHONE_NUMBER=SQLDA("SQLDA1", "SQLDATA", 10)
360 .WORK_DEPARTMENT=SQLDA("SQLDA1", "SQLDATA", 11)
370 .SEX=SQLDA("SQLDA1", "SQLDATA", 12)
380 EXIT

```

Delete program using static SQL statements

This program deletes one employee from an employee table using static SQL statements.

```
10 ENTRY STATIC_DELETE
20 .TEXT EMPLOYEE_NUMBER(6)
30 .EMPLOYEE_NUMBER "000120"
40 .EXEC_SQL
50 ..|
60 ..|   DELETE FROM DSN82.TEMPL
70 ..|
80 ..|   WHERE EMPNO = :EMPLOYEE_NUMBER
90 .END
100 EXIT
```

Delete program using dynamic SQL statements

This program deletes one employee from an employee table using dynamic SQL statements.

```
10 ENTRY DYNAMIC_DELETE
20 .TEXT EMPLOYEE NUMBER(6),SQL TEXT(254)
30 .EMPLOYEE_NUMBER "000120"
40 .SQL_TEXT="DELETE FROM DSN82.TEMPL WHERE EMPNO = ? "
50 .|
60 .EXEC_SQL
70 ..|
80 ..|   PREPARE S1 FROM :SQL_TEXT
90 ..|
100 .END
110 .EXEC_SQL
120 ..|
130 ..|   EXECUTE S1 USING :EMPLOYEE_NUMBER
140 ..|
150 .END
160 EXIT
```

Column select program using dynamic SQL statements

This program uses dynamic SQL statements to retrieve column names, data types, lengths, and the first row of the columns from a table specified by the user.

```

10 ENTRY SQL_LIST_TABLES
20 .|
30 .| THIS PROGRAM LISTS COLUMNS BASED ON TABLE NAME
40 .|
50 .TEXT TABLE_NAME(32)
60 .TEXT SQL_FUNCTION(100)
70 .SHOW"PLEASE ENTER TABLE NAME "
80 .OBTAIN TABLE_NAME
90 .SQL_FUNCTION="SELECT * FROM" +TABLE_NAME
100 .SQLDA("SQLDA1")=NEW
110 .EXEC_SQL
120 ..| PREPARE S1 FROM :SQL_FUNCTION
130 .END
140 .EXEC_SQL
150 ..| DESCRIBE S1 INTO SQLDA1
160 .END
170 .EXEC_SQL
180 ..| DECLARE C1 CURSOR FOR S1
190 .END
200 .EXEC_SQL
210 ..| OPEN C1
220 .END
230 .EXEC_SQL
240 ..| FETCH C1 USING DESCRIPTOR SQLDA1
250 .END
260 .COUNTER=1
270 .SHOW"COLUMN NAME",AT(45),"LENGTH",AT(55),"DATA"
280 .WHILE COUNTER<=SQLDA("SQLDA1" "SQLD")
290 ..SHOW SQLDA("SQLDA1", "SQLNAME", COUNTER),
300 ..'AT(25),TXT(SQLDA("SQLDA1", "SQLTYPE", COUNTER)),
310 ..'AT(45),TXT(SQLDA("SQLDA1", "SQLLEN", COUNTER)),
320 ..'AT(55), (SQLDA("SQLDA1", "SQLDATA", COUNTER))
330 ..COUNTER=COUNTER+1
340 .END
350 .WAIT
360 EXIT

```

Hold cursors across a COMMIT

This program shows how an SQL cursor can automatically be repositioned after an SQL COMMIT. You can do this by requesting that MANTIS return the value used in the search condition ("HOST_VAR1") as a host variable. When the cursor ("C1") is automatically closed by the CONVERSE statement (COMMIT at terminal I/O), the OPEN C1 statement will include all values for HOST_VAR1 that are greater than the last value for HOST_VAR retrieved before the COMMIT. In effect, this keeps the C1 cursor open across the COMMIT even though it has actually been closed and reopened. DB2 Version 2 Release 3 and later versions support the "WITH HOLD" clause of the DECLARE statement, which allows cursors to be held across a COMMIT statement. (Refer to the IBM DB2 documentation for further information.)

```
10 BIG HOST_VAR1,HOST_VAR2,HOST_VAR3
20 HOST_VAR1=ZERO
30 EXEC_SQL
40 .| DECLARE C1 CURSOR FOR
50 .| SELECT COL1, COL2, COL3
60 .| FROM TABLEA
70 .| WHERE COL1>:HOST_VAR1
80 END
90 WHILE SQLCA("SQLCODE")<>100
100 .EXEC_SQL:| OPEN C1
110 .END
120 .EXEC_SQL:| FETCH C1 INTO :HOST_VAR1, :HOST_VAR2, :HOST_VAR3
130 .END
140 .CONVERSE MAP1
150 END
```


B

Features not supported

The following features are not supported:

- ◆ The USING LABELS clause of the DESCRIBE statement is not implemented.
- ◆ Host variables may not be specified in a SELECT list. In the example below, the VX host variable is invalid:

```
SELECT A, :VX, C  
INTO :VA, :VB, :VC
```

- ◆ Exact line number reference when syntax errors are detected is not supported in all cases. Once control is transferred to the SQL database to execute an SQL statement, MANTIS no longer has control and therefore does not know on which line the error occurred. For example, if an error occurred in the INTO clause of the following statement:

```

01330 ..X=X+1
01340 ..EXEC_SQL
01350 ...|SELECT A,B,C
01360 ...|INTO :VA, :VB), :VC          <--- Error in this line
01370 ...|FROM TABLE.1
01380 ...|WHERE A=1
01390 ..END
01400 ..X=X-VA

```

MANTIS will point to the beginning of the SQL statement as being in error. For example:

```

01330 ..X=X+1
01340 ..EXEC_SQL
===>0 ...|SELECT A,B,C              <--- MANTIS points to this line
01360 ...|INTO :VA, :VB), :VC      <--- Error in this line
01370 ...|FROM TABLE.1
01380 ...|WHERE A=1
01390 ..END
01400 ..X=X-VA

```

- ◆ The contents of one SQLDA structure cannot be implicitly copied into another in a single instruction. The following statement is not permitted:

```
SQLDA( "NAME2" )=SQLDA( "NAME1" )
```

However, each element of an SQLDA can be moved individually to the corresponding element of a different SQLDA.

C

Comparing SQL in MANTIS SQL support to SQL in COBOL

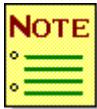
The following general considerations apply to SQL in MANTIS SQL Support as compared to SQL in COBOL:

- ◆ SQL statements are embedded in a MANTIS application program as MANTIS comments. Each SQL statement is bracketed with an EXEC_SQL-END block. No MANTIS comments are permitted within the EXEC_SQL-END block. All comments within the block are considered SQL statement text.
- ◆ In dynamic execution mode, a limited number of SQL statements can be active at one time. The Master User can modify this number; the default is 10, and the maximum is 512 for DB2 or 510 for DB2 for VSE and VM. This maximum applies to the MANTIS program and all DO levels within the program.
- ◆ In the SQL WHENEVER statement:
 - A MANTIS DO statement replaces the GOTO clause, and STOP is replaced by FAULT.
 - The default for the SQLERROR condition is FAULT; in SQL in COBOL, the default is CONTINUE.
 - WHENEVER settings may have different ranges of applicability than they would in SQL in COBOL.
- ◆ SQLCA elements are accessed through the SQLCA function and statement, rather than as items of data.
- ◆ SQLDA elements are accessed through the SQLDA function and statement, rather than as items of data.

- ◆ MANTIS SQL Support does not support the SQL INCLUDE statement. The SQLCA and SQLDA functions eliminate the need to INCLUDE these structures.
- ◆ SQL BEGIN DECLARE SECTION and END DECLARE SECTION statements are unnecessary. If you use these statements, MANTIS SQL Support regards them as comments.
- ◆ The SQL DECLARE TABLE statement is unnecessary. If you use this statement, MANTIS SQL Support regards it as a comment.
- ◆ In MANTIS, quotation marks (") delimit character-string constants. In SQL, apostrophes (') delimit character-string constants.
- ◆ SQL data types are supported in MANTIS compatible data types. These are listed in the following table.

Permissible data type conversions between SQL and MANTIS are listed in the following table. Note that this table is an exact replica of the table under “[Converting data between MANTIS SQL support and the SQL database](#)” on page 32; it is repeated here for convenience.

SQL data type	MANTIS data type	Considerations
DECIMAL	BIG, SMALL	Loss of precision may occur when converting from SQL to MANTIS. Overflow may occur when converting from MANTIS to SQL.
INTEGER	BIG, SMALL	Loss of precision may occur when converting from SQL to MANTIS. Overflow may occur when converting from MANTIS to SQL.
SMALLINT	BIG, SMALL	Overflow may occur when converting from MANTIS to SQL.
FLOAT	BIG, SMALL	Overflow and/or loss of precision may occur when converting from SQL to MANTIS SMALL variables.
CHAR VARCHAR LONG VARCHAR	TEXT	Truncation may occur in either direction.
DATE	TEXT	Truncation may occur if TEXT size is less than 10.
TIME	TEXT	Truncation may occur if TEXT size is less than 8.
TIMESTAMP	TEXT	Truncation may occur if TEXT size is less than 26.



The SQL data types LONG VARCHAR and LONG VARGRAPHIC are not fully supported in MANTIS SQL Support. They are treated in the same way as the SQL types VARCHAR and VARGRAPHIC: they are supported as TEXT or KANJI variables, having a maximum of 254 characters (TEXT) or 127 characters (KANJI).

- ◆ When dynamic SQL statements are used, only data type codes for MANTIS-compatible data types are returned in the SQLTYPE element in the SQLDA. Valid data types are thus limited to those listed in the following table. Note that the first data type code is returned if no indicator variable is present. The second data type code is returned when an indicator variable is present. Note that this table is an exact replica of the table under “[Move data from an SQLDA repeating group into a MANTIS program](#)” on page 69; it is repeated here for convenience.

SQL data type	Description	SQL type* type*	SQL type set by MANTIS	MANTIS type
DATE	Calendar date	384/385	448/449	TEXT
TIME	Time	388/389	448/449	TEXT
TIMESTAMP	Timestamp	392/393	448/449	TEXT
VARCHAR	Variable string	448/449	448/449	TEXT
CHAR	Fixed length string	452/453	448/449	TEXT
LONG VARCHAR	Long variable string	456/457	448/449	TEXT
VARGRAPHIC	Variable graphic string	464/465	464/465	KANJI
GRAPHIC	Fixed length graphic string	468/469	464/465	KANJI
LONG VARGRAPHIC	Long variable graphic string	472/473	464/465	KANJI
FLOAT	Floating point number	480/481	480/481	BIG
DECIMAL	Packed decimal number	484/485	480/481	BIG
INTEGER	Long integer	496/497	480/481	BIG
SMALLINT	Short integer	500/501	480/481	BIG

* The first number in the column is used when no indicator variable is present. This value does not allow NULL values. The second number is used when an indicator variable is present, and NULL data values can be specified.

D

SQL keywords

This appendix lists the SQL keywords used by MANTIS SQL Support. SQL keywords are not MANTIS reserved words and can be used in MANTIS SQL programs. However, if SQL keywords are used as host variable names in SQL statements, errors can occur. MANTIS SQL Support does limited parsing of the SQL statement text before passing the SQL statement to the database for execution. During parsing, host variables, which are the same as SQL keywords, may be taken as keywords instead of host variables causing a MANTIS fault. In the example below, MANTIS could display a fault message because “:COLUMN” is used as a host variable but is an SQL keyword.

```
.TEXT COLUMN(10), COLUMN1(20), COLUMN2(10)
.EXEC_SQL
..| SELECT COLUMN, COLUMN1, COLUMN2
..|   INTO :COLUMN, :COLUMN1, :COLUMN2
..|   FROM TABLEA
.END SQL Keywords
```

The following table lists the SQL keywords used by MANTIS SQL Support:

ALL	FOUND	RESET
AS	FROM	ROLLBACK
BEGIN	HOLD	SAME
BUFFER	IDENTIFIED	SEARCH
BY	IMMEDIATE	SECTION
CLOSE	INCLUDE	SELECT
COLUMN	INDEX	SERVER
COMMIT	INDEXNAME	SET
CONNECT	INSERT	SQLERROR
CONTINUE	INTO	SQL EXCEPTION
COPY	IS	SQLID
CURRENT	KEY	SQLWARNING
CURSOR	LAST	STATISTICS
DATE	LENGTH	TABLE
DBNAME	NEXT	TIME
DECLARE	NOT	TIMESTAMP
DELETE	OF	TIMEZONE
DESCRIBE	OPEN	TO
DESCRIPTOR	OUT	UNION
DO	PACKAGESET	UPDATE
DROP	POS	USER
END	PREPARE	USING
EXECUTE	PREV	VALUES
FAULT	PROGRAM	WHENEVER
FETCH	PUT	WHERE
FIRST	READ	WITH
FIRSTPOS	RELEASE	WORK
FOR		

Index

A

- Allocate an SQLDA 59
- Apostrophe (') 152
- Arrays 30, 72
- Authority - executing DB2 Application Plan 98

B

- Batch MANTIS 112
- Binding 51, 82
 - SQL binding 82
- Blanks 25

C

- Checking consistency of a bound program 104
- Checking the results of an SQL bind
 - extended dynamic execution mode 127
 - static execution mode 92
- CICS 50
- COBOL see SQL in COBOL
- Colon character 24
- Colon character () 18
- COMMIT WORK 50
- CONTINUE 42
- Converting data between MANTIS and SQL 32, 153
- Creating DB2 Application Plan 98
- Creating SQL Support Load Module 97
- Cursors 40

D

- Data type conversion 33, 153
- Data type specification 35
- DBTYPE 49
- Deallocate an SQLDA 60
- Declaring SQL cursors multiple times 79
- DO 42
- Dynamic execution mode 20, 151
- Dynamic statements 19, 55

E

- Embedding SQL statements in MANTIS programs 18, 151
- EMBEDDING SQL statements in MANTIS programs 23
- ENTRY statement parameters as host variables 54, 80
- Error messages 52
- EXEC_SQL-END 23
- Execution authority - DB2 Application Plan 98
- Execution modes 20
- Extended dynamic execution mode x, 21, 113

F

- FAULT 42
- Features not supported 149

G

- Generate SQL Support Source Module 93
 - creating 97
 - sample input 93

H

- High-Performance Option (HPO) 51
- Host variables 18, 24, 27, 34, 54
 - data types in extended dynamic execution mode 115
 - data types in static execution mode 78

I

Included SQL Support Load
Module 99, 101
Indicator variables 18, 31
static execution mode 78

K

Keywords 155

L

Large SQL statements 72

M

Maintaining SQL bind information
111
Making SQL Support Load
Module available to
MANTIS 99
MANTIS entities as host
variables 28
MANTIS SQL Support programs
in static execution mode 75
Move data from SQLDA
repeating group into
program 69
Move data into SQLDA repeating
group 64
MSGTEXT 49
Multiple lines 25
Multiple row result sets with
SELECT 81

N

NOT FOUND 41

P

Permanently loaded SQL
Support Load Module 99,
102
Prepare MANTIS SQL Support
programs for extended
dynamic execution mode
117
Prepare MANTIS SQL Support
programs for static
execution mode 82

Q

Question mark (?) 56
Quotation mark (") 152

R

Read header elements 67
ROLLBACK WORK 50
Row result sets with SELECT
extended dynamic execution
mode 116
static execution mode 81
Running a program from a line
number 51
Running SQL programs with
Batch MANTIS 112

S

- Sample batch input for SQL bind
 - extended dynamic execution mode 125
 - static execution mode 91, 126
- Sample programs 135
- Scope of SQL cursors and statements 40
- SELECT
 - extended dynamic execution mode 116
 - static execution mode 81
- Set SQLDA header information 61
- Spaces 25
- SQL bind (static execution mode)
 - checking results 92
- SQL binding (extended dynamic execution mode)
 - batch 125
 - check consistency 128
 - check results 127
 - online 118
 - unbinding 131
- SQL binding (static execution mode) 82
 - batch 90
 - checking consistency 104
 - maintaining information 110
 - online 83
 - unbinding 107
- SQL cursors and statements 40
 - static execution mode 79
- SQL data types 34, 152
- SQL in COBOL 17, 37, 151
- SQL keywords 155
- SQL statements
 - larger than 254 characters 72
 - limits 39
 - scope 40, 45
- SQL Support Load Module
 - available to MANTIS 99
- SQLABC 48
- SQLAID 48
- SQLCA 45
 - COBOL 151
 - elements 48
 - function 47
 - statement 45
- SQLCODE 48
- SQLD 63
- SQLDA 57
 - COBOL 151
 - header elements 57, 63
 - repeating elements 57
 - statement 59
 - structure 57
- SQLDABC 63
- SQLDAID 63
- SQLDATA 66
- SQLERRDn 48
- SQLERRM 48
- SQLERROR 41
- SQLERRP 48
- SQLEXT 48
- SQLIND 66
- SQLLEN 66
- SQLN 63
- SQLNAME 66
- SQLTYPE 66
 - codes 154
- SQLWARNING 41
- SQLWARNn 48
- Static execution mode 20, 75
 - authority - executing DB2
 - Application Plan 98
 - Batch MANTIS 112
 - checking consistency of a bound program 104
 - checking results of an SQL bind 92
 - creating DB2 Application Plan 98
 - creating SQL Support Load Module 97
 - declaring cursors multiple times 79
 - execution authority - DB2
 - Application Plan 98
 - generate SQL Support Source Module 93
 - creating 97
 - sample input 93
 - included SQL Support Load Module 99
 - maintaining SQL bind information 110
 - making SQL Support Load Module available to MANTIS 99
 - multiple row result sets with SELECT 81

Static execution mode (*cont.*)
 permanently loaded SQL
 Support Load Module 99,
 102
 preparing 82
 row result sets with SELECT'
 81
 running SQL programs with
 Batch MANTIS 112
 sample batch input for SQL
 bind 91, 126
 SELECT 81
 SQL binding 82
 batch 90
 check consistency 104
 check results 92
 maintaining information 110
 online 83
 unbinding 107
 SQL Support Load Module
 available to MANTIS 99
 temporarily loaded SQL
 Support Load Module 99,
 103
 unbinding SQL bound
 programs 107
 Static statements 19

T

Temporarily loaded SQL Support
 Load Module 99, 103
 Text literals 26

U

Unbinding SQL bound programs
 extended dynamic execution
 mode 131
 static execution mode 107

V

Variables See Host variables,
 Indicator variables, MANTIS
 variables
 Vertical bar (|) 18, 24

W

WHENEVER 40
 COBOL 151
 declarative statement 44
 defaults 43
 WHILE loop 44
 WORK See COMMIT WORK,
 ROLLBACK WORK