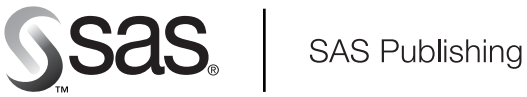


SAS/ACCESS® 9.1.2 Supplement for MySQL

SAS/ACCESS for Relational Databases

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *SAS/ACCESS® 9.1.2 Supplement for MySQL (SAS/ACCESS® for Relational Databases)*. Cary, NC: SAS Institute Inc.

SAS/ACCESS® 9.1.2 Supplement for MySQL (SAS/ACCESS® for Relational Databases)

ISBN 1-59047-458-9

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

1st printing, April 2004

SAS Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/pubs** or call 1-800-727-3228.

Contents

CHAPTER

1

SAS/ACCESS for MySQL

Introduction to the SAS/ACCESS Interface to MySQL

This document includes details about *only* the SAS/ACCESS Interface to MySQL. It should be used as a supplement to the main SAS/ACCESS documentation, *SAS/ACCESS for Relational Databases: Reference*.

LIBNAME Statement Specifics for MySQL

This section describes the LIBNAME statements as supported in the SAS/ACCESS interface to MySQL. For a complete description of this feature, see the LIBNAME statement section in *SAS/ACCESS for Relational Databases: Reference*. The MySQL specific syntax for the LIBNAME statement is as follows:

LIBNAME *libref* **mysql** <*connection-options*><*LIBNAME-options*>;

Arguments

libref
is any SAS name that serves as an alias to associate SAS with a database, schema, server, or group of tables.

mysql
is the SAS/ACCESS engine name for the interface to MySQL.

connection-options
provide connection information for the connection to the DBMS. The connection options for the interface to MySQL are:

USER=<'>*username*<'>
specifies the MySQL user login ID. If this argument is not specified, the current user is assumed. If the user name contains spaces or non-alphanumeric characters, you must enclose the user name in quotation marks.

PASSWORD=<'>*password*<'>
specifies the MySQL password that is associated with the MySQL login ID. If the password contains spaces or non-alphanumeric characters, you must enclose the password in quotation marks.

DATABASE=<'>*database*<'>
specifies the MySQL database to which you want to connect. If the database name contains spaces or non-alphanumeric characters, you must enclose the database name in quotation marks.

SERVER=<'>*server*<'>
specifies the server name or IP address of the MySQL server. If the server name contains spaces or non-alphanumeric characters, you must enclose the server name in quotation marks.

PORT=*port*
specifies the port used to connect to the specified MySQL server.

LIBNAME-options
define how DBMS objects are processed by SAS. Some LIBNAME options can enhance performance; others determine naming behavior. The following table describes LIBNAME options that are supported for MySQL and presents default values where applicable. See the section about the SAS/ACCESS LIBNAME statement in *SAS/ACCESS for Relational Databases: Reference* for detailed information about these options.

Table 1.1 SAS/ACCESS LIBNAME Options for MySQL

Option	Default Value
ACCESS=	NONE
AUTOCOMMIT=	YES
CONNECTION=	SHAREDREAD
CONNECTION_GROUP=	NONE
DBCOMMIT=	1000 when inserting rows; 0 when updating rows, deleting rows, or appending rows to an existing table

Option	Default Value
DBCONINIT=	NONE
DBCONTERM=	NONE
DBCREATE_TABLE_OPTS=	NONE
DBGEN_NAME=	DBMS
DBINDEX=	NO
DBLIBINIT=	NONE
DBLIBTERM=	NONE
DBMAX_TEXT=	1024
DBPROMPT=	NO
DBSASLABEL=	COMPAT
DEFER=	NO
DIRECT_EXE=	NONE
DIRECT_SQL=	YES
MULTI_DATASRC_OPT=	NONE
PRESERVE_COL_NAMES=	NO
PRESERVE_TAB_NAMES=	NO
QUALIFIER=	NONE
REREAD_EXPOSURE=	NO
SPOOL=	YES
UTILCONN_TRANSIENT=	NO

MySQL LIBNAME Statement Examples

In the following example, the libref MYSQLLIB uses the SAS/ACCESS interface to MySQL to connect to a MySQL database. The SAS/ACCESS connection options are USER=, PASSWORD=, DATABASE=, SERVER=, and PORT=.

```
libname mysqllib mysql user=testuser password=testpass database=mysqldb
server=mysqlserv port=9876;

proc print data=mysqllib.employees;
   where dept='CSR010';
run;
```

Data Set Options for MySQL

The following table describes all of the data set options that are supported for the MySQL interface. Default values are provided where applicable. See the section about data set options in *SAS/ACCESS for Relational Databases: Reference* for general information about these options.

Table 1.2 Data Set Options for MySQL

Option	Default Value
AUTOCOMMIT=	the current LIBNAME option setting
DBCOMMIT=	the current LIBNAME option setting
DBCONDITION=	none
DBCREATE_TABLE_OPTS=	the current LIBNAME option setting
DBGEN_NAME=	DBMS
DBINDEX=	the current LIBNAME option setting
DBKEY=	none
DBLABEL=	NO
DBMASTER=	none
DBMAX_TEXT=	1024
DBNULL=	YES
DBPROMPT=	the current LIBNAME option setting
DBSASLABEL=	COMPAT
DBSASTYPE=	See "Data Types for MySQL Servers" on page 9.
DBTYPE=	See "LIBNAME Statement Data Conversions" on page 11.
NULLCHAR=	SAS
NULLCHARVAL=	a blank character
PRESERVE_COL_NAMES=	current LIBNAME option setting
QUALIFIER=	the current LIBNAME option setting
SASDATEFORMAT=	DATETIME20.0
UPDATE_ISOLATION_LEVEL=	the current LIBNAME option setting

Pass-Through Facility Specifics for MySQL

See the section about the Pass-Through Facility in *SAS/ACCESS for Relational Databases: Reference* for general information about this feature.

The Pass-Through Facility specifics for MySQL are as follows:

- The *dbms-name* is **mysql**.
- The *database-connection-arguments* for the CONNECT statement are as follows:

  USER=<'>*MySQL-login-ID*<'>
  specifies an optional MySQL login ID. If USER= is not specified, the current user is assumed. If you specify USER=, you also must specify PASSWORD=.

  PASSWORD=<'>*MySQL-password*<'>
  specifies the MySQL password that is associated with the MySQL login ID. If you specify PASSWORD=, you also must specify USER=.

DATABASE=<'>*database-name*<'>
specifies the MySQL database.

SERVER=<'>*server-name*<'>
specifies the name or IP address of the MySQL server to which to connect. If *server-name* is omitted or set to **`localhost`**, a connection to the local host is established.

PORT=*port*
specifies the port on the server that is used for the TCP/IP connection.

Examples

The following example uses the alias DBCON for the DBMS connection (the connection alias is optional):

```
proc sql;
   connect to mysql as dbcon
       (user=testuser password=testpass server=mysqlserv
        database=mysqldb port=9876);
quit;
```

The following example connects to MySQL and sends it two EXECUTE statements to process:

```
proc sql;
   connect to mysql (user=testuser password=testpass server=mysqlserv
        database=mysqldb port=9876);
   execute (create table whotookorders as
      select ordernum, takenby,
             firstname, lastname, phone
        from orders, employees
        where orders.takenby=employees.empid)
      by mysql;
   execute (grant select on whotookorders
            to testuser) by mysql;
   disconnect from mysql;
quit;
```

The following example performs a query, shown in highlighted text, on the MySQL table CUSTOMERS:

```
proc sql;
connect to mysql (user=testuser password=testpass server=mysqlserv
        database=mysqldb port=9876);
select *
   from connection to mysql
     (select * from customers
      where customer like '1%');
    disconnect from mysql;
quit;
```

Autocommit and Table Types

MySQL supports several table types, two of which are MyISAM (the default) and INNODB. A single database can contain tables of different types. The behavior of a

table is determined by its table type. For example, by definition, a table created of MyISAM type does not support transactions. Consequently, all DML statements (updates, deletes, inserts) are automatically committed. If you need transactional support, specify a table type of INNODB in the DBCREATE_TABLE_OPTS LIBNAME option. This table type allows for updates, deletes, and inserts to be rolled back if an error occurs; or updates, deletes, and inserts to be committed if the SAS DATA step or procedure completes successfully.

By default, the MYSQL libname engine sets **AUTOCOMMIT=YES** regardless of the table type. If you are using tables of the type INNODB, set the LIBNAME option AUTOCOMMIT=NO to improve performance. To control how often COMMITS are executed, set the DBCOMMIT option.

Note: The DBCOMMIT option can affect SAS/ACCESS performance. Experiment with a value that best fits your table size and performance needs before using it for production jobs. Transactional tables require significantly more memory and disk space requirements. △

Understanding MySQL Update and Delete Rules

To avoid data integrity problems when updating or deleting data, you need a primary key defined on your table. Refer to the MySQL documentation for more information regarding table types and transactions.

The following example uses AUTOCOMMIT=NO and DBTYPE to create the primary key, and DBCREATE_TABLE_OPTS to determine the MySQL table type.

```
libname invty mysql user=dbitest server=d6687 database=test autocommit=no
reread_exposure=no;

proc sql;
drop table invty.STOCK23;
quit;

/* Create DBMS table with primary key and of type INNODB*/
data invty.STOCK23(drop=PARTNO DBTYPE=(RECDATE="date not null,
                    primary key(RECDATE)") DBCREATE_TABLE_OPTS="type = innodb");
  input PARTNO $ DESCX $ INSTOCK @17
        RECDATE date7. @25 PRICE;
  format RECDATE date7.;
  datalines;
K89R seal  34   27jul95 245.00
M447 sander 98  20jun95 45.88
LK43 filter 121 19may96 10.99
MN21 brace 43   10aug96 27.87
BC85 clamp 80   16aug96 9.55
KJ66 cutter 6   20mar96 24.50
UYN7 rod  211   18jun96 19.77
JD03 switch 383 09jan97 13.99
BV1I timer 26   03jan97 34.50
;

proc sql;
update invty.STOCK23 set price=price*1.1 where INSTOCK > 50;
quit;
```

Passing SAS Functions to MySQL

The interface to MySQL passes the following SAS functions to MySQL for processing. Where the MySQL function name differs from the SAS function name, the MySQL name appears in parentheses. See "Passing Functions to the DBMS using PROC SQL" in *SAS/ACCESS for Relational Databases: Reference* for information.

ABS

ARCOS (ACOS)

ARSIN (ASIN)

ATAN

BYTE

CEIL (CEILING)

COMPRESS

COS

COT

DATE

DATETIME

DAY

EXP

FLOOR

HOUR

INDEX

LOWCASE (LCASE)

LENGTH

LOG

LOG10

MINUTE

MOD

MONTH

QTR

REPEAT

SECOND

SIGN

SIN

SOUNDEX

SQRT

SUBSTR

TAN

TIME

TODAY

TRIM (TRIMN)

UPCASE (UCASE)

WEEKDAY

YEAR

Passing Joins to MySQL

In order for a multiple libref join to pass to MySQL, all of the following components of the LIBNAME statements must match exactly:

user

password

database

server

See "Passing Joins to the DBMS" in *SAS/ACCESS for Relational Databases: Reference* for more information about when and how SAS/ACCESS passes joins to the DBMS.

Naming Conventions for MySQL

MySQL database identifiers that can be named include databases, tables, and columns. The MySQL documentation contains extensive on naming conventions. The following are some of the naming conventions that you must use.

- All identifier names must be from 1 to 64 characters long, except for aliases, which may be 255 characters.
- Database names must be unique. For each user within a database, names of database objects must be unique across all users (for example, if a database contains a department table created by user A, no other user can create a department table in the same database).

  Note: MySQL does not recognize the notion of schema. Consequently, tables are automatically visible to all users with appropriate privileges. Column names and index names must be unique within a table. △
- Database names can use any character that is allowed in a directory name except for periods and backward and forward slashes.
- Table names may use any character allowed in a filename except for periods and forward slashes.
- Column and alias names allow all characters.
- A name cannot be a MySQL reserved word unless the name is enclosed in quotation marks. See the MySQL documentation for more information about reserved words.
- Embedded spaces and other special characters are not permitted unless the name is enclosed in quotation marks.

- Embedded quotation marks are not permitted.
- Case sensitivity is set when a server is installed. By default, the names of database objects are case sensitive on UNIX and not case sensitive on Windows. For example, the names **CUSTOMER** and **customer** are different on a case-sensitive server.

Note: By default, column and table names are not quoted in the SAS/ACCESS interface to MySQL. To quote the table and column names, you must use the LIBNAME statement PRESERVE_TAB_NAMES=. △

Case Sensitivity for MySQL

In MySQL, databases and tables correspond to directories and files within those directories. Consequently, the case sensitivity of the underlying operating system determines the case sensitivity of database and table names. This means database and table names are case insensitive in Windows, and case sensitive in most varieties of UNIX.

In SAS, names can be entered in either uppercase or lowercase. MySQL recommends that you adopt a consistent convention of either all uppercase or all lowercase tablenames, especially on UNIX hosts. This can be easily implemented by starting your server with *-O lower_case_table_names=1*. Please see the MySQL documentation for more details.

If your server is on a case-sensitive platform, and you choose to allow case sensitivity, be aware that when you reference MYSQL objects through the SAS/ACCESS interface, objects are case sensitive and require no quotation marks. Furthermore, in the pass-through facility, all MySQL object names are case sensitive. The names are passed to MySQL exactly as they are typed.

For more information about case sensitivity and MySQL names, see Naming Conventions for MySQL.

Data Types for MySQL Servers

Overview

Every column in a table has a name and a data type. The data type tells MySQL how much physical storage to set aside for the column and the form in which the data is stored.

Character Data

BLOB
: contains binary data of variable length up to 64 kilobytes. Variables entered into columns of this type must be inserted as character strings.

CHAR *(n)*
: contains fixed-length character string data with a length of *n*, where *n* must be at least 1 and cannot exceed 255 characters.

ENUM (*"value1"*, *"value2"*, *"value3"*,...)
contains a character value that can be chosen from the list of allowed values. You can specify up to 65535 ENUM values. If the column contains a string not specified in the value list, the column value is set to "0".

LONGBLOB
contains binary data of variable length up to 4 gigabytes. Variables entered into columns of this type must be inserted as character strings. Available memory considerations might limit the size of a LONGBLOB data type.

LONGTEXT
contains text data of variable length up to 4 gigabytes. Available memory considerations might limit the size of a LONGTEXT data type.

MEDIUMBLOB
contains binary data of variable length up to 16 megabytes. Variables entered into columns of this type must be inserted as character strings.

MEDIUMTEXT
contains text data of variable length up to 16 megabytes.

SET (*"value1"*, *"value2*", *"value3"*,...)
contains zero or more character values that must be chosen from the list of allowed values. You can specify up to 64 SET values.

TEXT
contains text data of variable length up to 64 kilobytes.

TINYBLOB
contains binary data of variable length up to 256 bytes. Variables entered into columns of this type must be inserted as character strings.

TINYTEXT
contains text data of variable length up to 256 bytes.

VARCHAR (*n*)
contains character string data with a length of *n*, where *n* is a value from 1 to 255.

Numeric Data

BIGINT (*n*)
specifies an integer value, where *n* indicates the display width for the data. You might experience problems with MySQL if the data column contains values that are larger than the value of *n*. Values for BIGINT can range from -9223372036854775808 to 9223372036854775808.

DECIMAL (*length*, *decimals*)
specifies a fixed-point decimal number, where *length* is the total number of digits (precision), and *decimals* is the number of digits to the right of the decimal point (scale).

DOUBLE (*length*, *decimals*)
specifies a double-precision decimal number, where *length* is the total number of digits (precision), and *decimals* is the number of digits to the right of the decimal point (scale). Values can range from approximately –1.8E308 to –2.2E-308 and 2.2E-308 to 1.8E308 (if UNSIGNED is specified).

FLOAT (*length*, *decimals*)
specifies a floating-point decimal number, where *length* is the total number of digits (precision) and *decimals* is the number of digits to the right of the decimal

point (scale). Values can range from approximately –3.4E38 to –1.17E-38 and 1.17E-38 to 3.4E38 (if UNSIGNED is specified).

INT (*n*)
specifies an integer value, where *n* indicates the display width for the data. You might experience problems with MySQL if the data column contains values that are larger than the value of *n*. Values for INT can range from –2147483648 to 2147483647.

MEDIUMINT (*n*)
specifies an integer value, where *n* indicates the display width for the data. You might experience problems with MySQL if the data column contains values that are larger than the value of *n*. Values for MEDIUMINT can range from –8388608 to 8388607.

SMALLINT (*n*)
specifies an integer value, where *n* indicates the display width for the data. You might experience problems with MySQL if the data column contains values that are larger than the value of *n*. Values for SMALLINT can range from –32768 to 32767.

TINYINT (*n*)
specifies an integer value, where *n* indicates the display width for the data. You might experience problems with MySQL if the data column contains values that are larger than the value of *n*. Values for TINYINT can range from -128 to 127.

Other Data Types

DATE
contains date values. Valid dates are from January 1, 1000, to December 31, 9999. The default format is YYYY-MM-DD, for example, 1961–06–13.

DATETIME
contains date and time values. Valid values are from 00:00:00 on January 1, 1000, to 23:59:59 on December 31, 9999. The default format is YYYY-MM-DD HH:MM:SS, for example, 1992–09–20 18:20:27.

TIME
contains time values. Valid times are –838 hours, 59 minutes, 59 seconds to 838 hours, 59 minutes, 59 seconds. The default format is HH:MM:SS, for example, 12:17:23.

TIMESTAMP
contains date and time values used to mark data operations. Valid values are from 00:00:00 on January 1, 1970, to 2037. The default format is YYYY-MM-DD HH:MM:SS, for example, 1995–08–09 15:12:27.

LIBNAME Statement Data Conversions

The following table shows the default SAS variable formats that SAS/ACCESS assigns to MySQL data types during input operations when you use the LIBNAME statement.

Table 1.3 LIBNAME Statement: Default SAS Formats for MySQL Data Types

MySQL Column Type	SAS Data Type	Default SAS Format
CHAR(*n*)	character	$*n*.
VARCHAR(*n*)	character	$*n*.
TINYTEXT	character	$*n*.
TEXT	character	$*n*. (where *n* is the value of the DBMAX_TEXT= option)
MEDIUMTEXT	character	$*n*. (where *n* is the value of the DBMAX_TEXT= option)
LONGTEXT	character	$*n*. (where *n* is the value of the DBMAX_TEXT= option)
TINYBLOB	character	$*n*. (where *n* is the value of the DBMAX_TEXT= option)
BLOB	character	$*n*. (where *n* is the value of the DBMAX_TEXT= option)
MEDIUMBLOB	character	$*n*. (where *n* is the value of the DBMAX_TEXT= option)
LONGBLOB	character	$*n*. (where *n* is the value of the DBMAX_TEXT= option)
ENUM	character	$*n*.
SET	character	$*n*.
TINYINT	numeric	4.0
SMALLINT	numeric	6.0
MEDIUMINT	numeric	8.0
INT	numeric	11.0
BIGINT	numeric	20.
DECIMAL	numeric	*m.n*
FLOAT	numeric	
DOUBLE	numeric	
DATE	numeric	DATE
TIME	numeric	TIME
DATETIME	numeric	DATETIME
TIMESTAMP	numeric	DATETIME

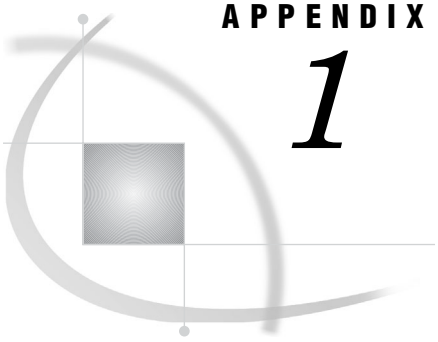
The following table shows the default MySQL data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

Table 1.4 LIBNAME Statement: Default MySQL Data Types for SAS Variable Formats

SAS Variable Format	MySQL Data Type
*m.n**	DECIMAL ([*m*-1],*n*)**
n (where *n* <= 2)	TINYINT
n (where *n* <= 4)	SMALLINT
n (where *n* <=6)	MEDIUMINT
n (where *n* <= 17)	BIGINT
other numerics	DOUBLE
$*n* (where *n* <= 255)	VARCHAR(*n*)
$*n* (where *n* > 255)	TEXT
datetime formats	TIMESTAMP
date formats	DATE
time formats	TIME

* *n* in MySQL data types is equivalent to *w* in SAS formats.

** DECIMAL types are created as (m-1, n). SAS includes space to write the value, the decimal point, and a minus sign (if necessary) in its calculation for precision These must be removed when converting to MySQL.



APPENDIX

1

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *SAS/ACCESS Supplement for DB2 under z/OS*
- *SAS/ACCESS Supplement for DB2 under UNIX and PC Hosts*
- *SAS/ACCESS Supplement for Informix*
- *SAS/ACCESS Supplement for Microsoft SQL Server*
- *SAS/ACCESS Supplement for ODBC*
- *SAS/ACCESS Supplement for OLE DB*
- *SAS/ACCESS Supplement for Oracle*
- *SAS/ACCESS Supplement for SYBASE*
- *SAS/ACCESS Supplement for Teradata*
- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *Base SAS Procedures Guide*
- SAS Companion that is specific to your operating environment

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: sasbook@sas.com
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Glossary

This glossary defines SAS software terms that are used in this document as well as terms that relate specifically to SAS/ACCESS software.

access descriptor
a SAS/ACCESS file that describes data that is managed by a data management system. After creating an access descriptor, you can use it as the basis for creating one or more view descriptors. See also view and view descriptor.

browsing data
the process of viewing the contents of a file. Depending on how the file is accessed, you can view SAS data either one observation (row) at a time or as a group in a tabular format. You cannot update data that you are browsing.

bulk load
to load large amounts of data into a database object, using methods that are specific to a particular DBMS. Bulk loading enables you to rapidly and efficiently add multiple rows of data to a table as a single unit.

client
(1) a computer or application that requests services, data, or other resources from a server. (2) in the X Window System, an application program that interacts with the X server and can perform tasks such as terminal emulation or window management. For example, SAS is a client because it requests windows to be created, results to be displayed, and so on.

column
in relational databases, a vertical component of a table. Each column has a unique name, contains data of a specific type, and has certain attributes. A column is analogous to a variable in SAS terminology.

column function
an operation that is performed for each value in the column that is named as an argument of the function. For example, AVG(SALARY) is a column function.

commit
the process that ends a transaction and makes permanent any changes to the database that the user made during the transaction. When the commit process occurs, locks on the database are released so that other applications can access the changed data. The SQL COMMIT statement initiates the commit process.

DATA step view
a type of SAS data set that consists of a stored DATA step program. Like other SAS data views, a DATA step view contains a definition of data that is stored elsewhere; the view does not contain the physical data. The view's input data can come from one or more sources, including external files and other SAS data sets. Because a DATA step view only reads (opens for input) other files, you cannot update the view's underlying data.

data type
a unit of character or numeric information in a SAS data set. A data value represents one variable in an observation.

data value
in SAS, a unit of character or numeric information in a SAS data set. A data value represents one variable in an observation.

database
an organized collection of related data. A database usually contains named files, named objects, or other named entities such as tables, views, and indexes

database management system (DBMS)
an organized collection of related data. A database usually contains named files, named objects, or other named entities such as tables, views, and indexes

editing data
the process of viewing the contents of a file with the intent and the ability to change those contents. Depending on how the file is accessed, you can view the data either one observation at a time or in a tabular format.

engine
a component of SAS software that reads from or writes to a file. Each engine enables SAS to access files that are in a particular format. There are several types of engines.

file
a collection of related records that are treated as a unit. SAS files are processed and controlled by SAS and are stored in SAS data libraries.

format
a collection of related records that are treated as a unit. SAS files are processed and controlled by SAS and are stored in SAS data libraries. In SAS/ACCESS software, the default formats vary according to the interface product.

index
(1) in SAS software, a component of a SAS data set that enables SAS to access observations in the SAS data set quickly and efficiently. The purpose of SAS indexes is to optimize WHERE-clause processing and to facilitate BY-group processing. (2) in other software vendors' databases, a named object that directs the DBMS to the storage location of a particular data value for a particular column. Some DBMSs have additional specifications. These indexes are also used to optimize the processing of WHERE clauses and joins. Depending on the SAS interface to a database product and how selection criteria are specified, SAS may or may not be able to use the indexes of the DBMS to speed data retrieval.

Depending on how selection criteria are specified, SAS might use DBMS indices to speed data retrieval.

informat
a pattern or set of instructions that SAS uses to determine how data values in an input file should be interpreted. SAS provides a set of standard informats and also enables you to define your own informats.

interface view engine
a SAS engine that is used by SAS/ACCESS software to retrieve data from files that have been formatted by another vendor's software. Each SAS/ACCESS interface has its own interface view engine, which reads the interface product data and returns the data in a form that SAS can understand (that is, in a SAS data set). SAS automatically uses an interface view engine; the engine name is stored in SAS/ACCESS descriptor files so that you do not need to specify the engine name in a LIBNAME statement.

libref
a name that is temporarily associated with a SAS data library. The complete name of a SAS file consists of two words, separated by a period. The libref, which is the first word, indicates the library. The second word is the name of the specific SAS file. For example, in VLIB.NEWBDAY, the libref VLIB tells SAS which library contains the file NEWBDAY. You assign a libref with a LIBNAME statement or with an operating system command.

member
a SAS file in a SAS data library.

member name
a name that is given to a SAS file in a SAS data library.

member type
a SAS name that identifies the type of information that is stored in a SAS file. Member types include ACCESS, DATA, CATALOG, PROGRAM, and VIEW.

missing value
in SAS, a term that describes the contents of a variable that contains no data for a particular row or observation. By default, SAS prints or displays a missing numeric value as a single period, and it prints or displays a missing character value as a blank space.

observation
a row in a SAS data set. All of the data values in an observation are associated with a single entity such as a customer or a state. Each observation contains one data value for each variable. In a database product table, an observation is analogous to a row. Unlike rows in a database product table or file, observations in a SAS data file have an inherent order.

Pass-Through Facility
a group of SQL procedure statements that send and receive data directly between a relational database management system and SAS. The Pass-Through Facility includes the CONNECT, DISCONNECT, and EXECUTE statements, and the CONNECTION TO component. SAS/ACCESS software is required in order to use the Pass-Through Facility.

PROC SQL view
a SAS data set (of type VIEW) that is created by the SQL procedure. A PROC SQL view contains no data. Instead, it stores information that enables it to read data values from other files, which can include SAS data files, SAS/ACCESS views, DATA step views, or other PROC SQL views. A PROC SQL view's output can be either a subset or a superset of one or more files.

query
a set of instructions that requests particular information from one or more data sources.

referential integrity
a set of rules that a DBMS uses to ensure that whenever a data value in one table is changed, the appropriate change is also made to any related values in other tables or in the same table. Referential integrity is also used to ensure that related data is not deleted or changed accidentally.

relational database management system
a database management system that organizes and accesses data according to relationships between data items. Oracle and DB2 are examples of relational database management systems.

rollback
in most databases, the process that restores the database to its state when changes were last committed, voiding any recent changes. The SQL ROLLBACK statement initiates the rollback processes. See also commit.

row
in relational database management systems, the horizontal component of a table. A row is analogous to a SAS observation.

SAS data file
a type of SAS data set that contains data values as well as descriptor information that is associated with the data. The descriptor information includes information such as the data types and lengths of the variables, as well as the name of the engine that was used to create the data. A PROC SQL table is a SAS data file. SAS data files are of member type DATA.

SAS data library
a collection of one or more SAS files that are recognized by SAS and that are referenced and stored as a unit. Each file is a member of the library.

SAS data set
a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views. SAS data files contain data values in addition to descriptor information that is associated with the data. SAS data views contain only the descriptor information plus other information that is required for retrieving data values from other SAS data sets or from files whose contents are in other software vendors' file formats.

SAS data view
a file whose contents are in one of the native SAS file formats. There are two types of SAS data sets: SAS data files and SAS data views. SAS data files contain data values in addition to descriptor information that is associated with the data. SAS data views contain only the descriptor information plus other information that is required for retrieving data values from other SAS data sets or from files whose contents are in other software vendors' file formats.

SAS/ACCESS views
See view descriptor and SAS data view.

server
in a network, a computer that is reserved for servicing other computers in the network. Servers can provide several different types of services, such as file services and communication services. Servers can also enable users to access shared resources such as disks, data, and modems.

Structured Query Language (SQL)
the standardized, high-level query language that is used in relational database management systems to create and manipulate database management system objects. SAS implements SQL through the SQL procedure.

table
a two-dimensional representation of data, in which the data values are arranged in rows and columns.

trigger
a type of user-defined stored procedure that is executed whenever a user issues a data-modification command such as INSERT, DELETE, or UPDATE for a specified table or column. Triggers can be used to implement referential integrity or to maintain business constraints.

variable
a column in a SAS data set. A variable is a set of data values that describe a given characteristic across all observations.

view
a definition of a virtual data set. The definition is named and stored for later use. A view contains no data; it merely describes or defines data that is stored elsewhere. SAS data views can be created by the ACCESS and SQL procedures.

view descriptor
a file created by SAS/ACCESS software that defines part or all of the database management system (DBMS) data or PC file data that is described by an access descriptor. The access descriptor describes the data in a single DBMS table, DBMS view, or PC file.

wildcard
a file created by SAS/ACCESS software that defines part or all of the database management system (DBMS) data or PC file data that is described by an access descriptor. The access descriptor describes the data in a single DBMS table, DBMS view, or PC file.

Index

Your Turn

If you have comments or suggestions about *SAS/ACCESS 9.1.2 Supplement for MySQL (SAS/ACCESS for Relational Databases)*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **yourturn@sas.com**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **suggest@sas.com**