



THE  
POWER  
TO KNOW®



# **SAS/ACCESS® 9.1.3**

## **Supplement for Netezza**

SAS/ACCESS for Relational Databases

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2007. *SAS/ACCESS® 9.1.3 Supplement for Netezza (SAS/ACCESS for Relational Databases)*. Cary, NC: SAS Institute Inc.

**SAS/ACCESS® 9.1.3 Supplement for Netezza (SAS/ACCESS for Relational Databases)**

Copyright © 2002–2007, SAS Institute Inc., Cary, NC, USA

ISBN 978-1-59994-277-3

All rights reserved. Produced in the United States of America.

**For a hard-copy book:** No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**For a Web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

**U.S. Government Restricted Rights Notice.** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

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

1st printing, August 2007

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.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

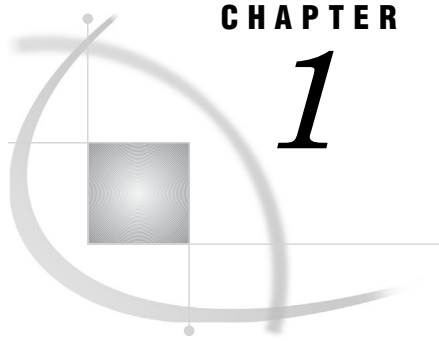
Other brand and product names are registered trademarks or trademarks of their respective companies.

---

# Contents

<b>Chapter 1</b>	<b>△ SAS/ACCESS for Netezza</b>	<b>1</b>
Introduction to SAS/ACCESS Interface to Netezza		1
LIBNAME Statement Specifics for Netezza		2
Data Set Options for Netezza		5
Pass-Through Facility Specifics for Netezza		6
Passing SAS Functions to Netezza		8
Passing Joins to Netezza		9
Temporary Table Support for Netezza		10
Netezza Bulk Loading and Unloading		11
Naming Conventions for Netezza		13
Data Types for Netezza		14
<b>Appendix 1</b>	<b>△ Recommended Reading</b>	<b>19</b>
Recommended Reading		19
<b>Glossary</b>		<b>21</b>
<b>Index</b>		<b>27</b>





## CHAPTER

## 1

# SAS/ACCESS for Netezza

---

<i>Introduction to SAS/ACCESS Interface to Netezza</i>	1
<i>LIBNAME Statement Specifics for Netezza</i>	2
Overview	2
Arguments	2
Netezza LIBNAME Statement Examples	4
<i>Data Set Options for Netezza</i>	5
Overview	5
<i>Pass-Through Facility Specifics for Netezza</i>	6
Overview	6
CONNECT Statement Examples	6
Special Catalog Queries	7
<i>Passing SAS Functions to Netezza</i>	8
<i>Passing Joins to Netezza</i>	9
<i>Temporary Table Support for Netezza</i>	10
General Information	10
Establishing a Temporary Table	10
Terminating a Temporary Table	10
Examples	10
<i>Netezza Bulk Loading and Unloading</i>	11
Loading	11
Examples	12
Unloading	12
Examples	13
<i>Naming Conventions for Netezza</i>	13
<i>Data Types for Netezza</i>	14
Overview	14
String Data	14
Numeric Data	14
Dates, Times, and Timestamps	15
Netezza Null Values	15
LIBNAME Statement Data Conversions	16

---

## Introduction to SAS/ACCESS Interface to Netezza

This document describes *only* SAS/ACCESS Interface to Netezza. Use it as a supplement to the generic SAS/ACCESS documentation, *SAS/ACCESS for Relational Databases: Reference*.

## LIBNAME Statement Specifics for Netezza

---

### Overview

This section describes the LIBNAME statement that SAS/ACCESS Interface to Netezza supports. For a complete description of this feature, see the LIBNAME statement section in *SAS/ACCESS for Relational Databases: Reference*. Here is the Netezza-specific syntax for the LIBNAME statement:

```
LIBNAME libref netezza <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 and views.

*netezza*

is the SAS/ACCESS engine name for SAS/ACCESS Interface to Netezza.

*connection-options*

provide connection information and control how SAS manages the timing and concurrence of the connection to the DBMS. When you use the LIBNAME statement, you can connect to the Netezza Performance Server in two ways. Use *only one* of the following methods for each connection because they are mutually exclusive.

- specify SERVER=, DATABASE=, PORT=, USER=, PASSWORD=, and READ\_ONLY=, or
- specify DSN=, USER=, and PORT=

Here is how these connection options are defined.

**SERVER=<'>***server-name***<'>**

specifies the server name or IP address of the Netezza Performance Server to which you want to connect. This server accesses the database that contains the tables and views that you want to access. If the server name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

**DATABASE=<'>***database-name***<'>**

specifies the name of the database on the Netezza Performance Server that contains the tables and views that you want to access. If the database name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks. You can also specify DATABASE= with the DB= alias.

**PORT=***port*

specifies the port number that is used to connect to the specified Netezza Performance Server. If you do not specify a port, the default of 5480 is used.

**USER=<'>***Netezza-user-name***<'>**

specifies the Netezza user name (also called the user ID) that you use to connect to your database. If the user name contains spaces or nonalphanumeric characters, you must enclose it in quotation marks.

PASSWORD=<'>Netezza-password<'>

specifies the password that is associated with your Netezza user name. If the password contains spaces or nonalphanumeric characters, you must enclose it in quotation marks. You can also specify PASSWORD= with the PWD=, PASS=, and PW= aliases.

READ\_ONLY=YES | NO

specifies whether to connect to the Netezza database in read-only mode (YES) or read-write (NO) mode. If you do not specify anything for READ\_ONLY=, the default of NO is used. You can also specify READ\_ONLY with the READONLY= alias.

DSN=<'>Netezza-data-source<'>

specifies the configured Netezza ODBC datasource to which you want to connect. Use this option if you have existing Netezza ODBC datasources that are configured on your client. This method requires additional setup—either through the ODBC Administrator control panel on Windows platforms or through the odbc.ini file on UNIX platforms. So it is recommended that you use this connection method only if you have existing, functioning datasources that have been defined.

#### *LIBNAME-options*

define how SAS processes DBMS objects. Some LIBNAME options can enhance performance; others determine locking or naming behavior. The following table describes the LIBNAME options that are supported for Netezza 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 Netezza

Option	Default Value
ACCESS=	none
AUTOCOMMIT=	operation-specific
CONNECTION=	UNIQUE
CONNECTION_GROUP=	none
DBCOMMIT=	1000 (inserting) or 0 (updating)
DBCONINIT=	none
DBCONTERM=	none
DB_CREATE_TABLE_OPTS=	none
DBGEN_NAME=	DBMS
DBINDEX=	YES
DBLIBINIT=	none
DBLIBTERM=	none
DBMAX_TEXT=	1024
DBNULLKEYS=	YES
DBPROMPT=	NO
DEFER=	NO
DELETE_MULT_ROWS=	

Option	Default Value
DIRECT_EXE=	none
DIRECT_SQL=	YES
IGNORE_READ_ONLY_COLUMNS=	NO
INSERTBUFF=	automatically calculated based on row length
MULTI_DATASRC_OPT=	none
PRESERVE_COL_NAMES=	see Naming Conventions for Netezza“Naming Conventions for Netezza” on page 13
PRESERVE_TAB_NAMES=	see Naming Conventions for Netezza“Naming Conventions for Netezza” on page 13
QUALIFIER=	none
QUERY_TIMEOUT=	0
QUOTE_CHAR=	none
READBUFF=	automatically calculated based on row length
REREAD_EXPOSURE=	NO
SCHEMA=	none
SPOOL=	YES
SQL_FUNCTIONS=	none
STRINGDATES=	NO
TRACE=	NO
TRACEFILE=	none
UPDATE_MULT_ROWS=	
USE_ODBC_CL =	NO
UTILCONN_TRANSIENT=	NO

---

## Netezza LIBNAME Statement Examples

In this example, SERVER=, DATABASE=, USER=, and PASSWORD= are connection options.

```
libname mydblib netezza server=npssrv1 database=test user=netusr1 password=netpwd1;

proc print data=mydblib.customers;
    where state='CA';
run;
```



In the next example, DSN=, USER=, and PASSWORD= are connection options. The NZSQL data source is configured in the ODBC Administrator Control Panel on Windows platforms or in the odbc.ini file—or a similarly named configuration file—on UNIX platforms.

```
libname mydblib netezza dsn=NZSQL user=netusr1 password=netpwd1;

roc print data=mydblib.customers;
    where state='CA';
run;
```

## Data Set Options for Netezza

### Overview

The following table describes the data set options that are supported for Netezza and provides default values where applicable. See the section about data set options in *SAS/ACCESS for Relational Databases: Reference* for detailed information about these options.

**Table 1.2** SAS/ACCESS Data Set Options

Option	Default Value
BL_DATAFILE=	When BL_USE_PIPE=NO, creates a file in the current directory or with the default file specifications.
BL_DELETE_DATAFILE=	YES (only when BL_USE_PIPE=NO)
BL_DELIMITER=	(the pipe symbol)
BL_OPTIONS=	none
BL_USE_PIPE=	YES
BULKLOAD=	NO
BULKUNLOAD=	NO
DBCOMMIT=	LIBNAME option setting
DBCONDITION=	none
DBCREATE_TABLE_OPTS=	LIBNAME option setting
DBFORCE=	NO
DBGEN_NAME=	DBMS
DBINDEX=	LIBNAME option setting
DBKEY=	none
DBLABEL=	NO
DBMASTER=	none
DBMAX_TEXT=	1024

Option	Default Value
DBNULL=	YES
DBNULLKEYS=	LIBNAME option setting
DBPROMPT=	LIBNAME option setting
DBSASTYPE=	see Data Types for Netezza“Data Types for Netezza” on page 14
DBTYPE=	see Data Types for Netezza“Data Types for Netezza” on page 14
ERRLIMIT=	1
IGNORE_READ_ONLY_COLUMNS=	NO
INSERTBUFF=	LIBNAME option setting
NULLCHAR=	SAS
NULLCHARVAL=	a blank character
PRESERVE_COL_NAMES=	LIBNAME option setting
QUALIFIER=	LIBNAME option setting
QUERY_TIMEOUT=	LIBNAME option setting
READBUFF=	LIBNAME option setting
SASDATEFMT=	none
SCHEMA=	LIBNAME option setting

---

## Pass-Through Facility Specifics for Netezza

---

### Overview

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

Here are the Pass-Through Facility specifics for the Netezza interface.

- The *dbms-name* is **NETEZZA**.
- The CONNECT statement is required.
- PROC SQL supports multiple connections to Netezza. If you use multiple simultaneous connections, you must use the *alias* argument to identify the different connections. If you do not specify an alias, the default **netezza** alias is used.
- The CONNECT statement *database-connection-arguments* are identical to its LIBNAME connection-options.

---

### CONNECT Statement Example

This example uses the DBCON alias to connection to the **mynpssrv** Netezza Performance Server and execute a query. The connection alias is optional.

```
proc sql;
  connect to netezza as dbcon
    (server=mynpssrv database=test user=myuser password=mypwd);
select * from connection to dbcon
  (select * from customers where customer like '1%');
quit;
```

---

## Special Catalog Queries

SAS/ACCESS Interface to Netezza supports the following special queries, which you can use to call the ODBC-style catalog function APIs. Here is the general format of the special queries.

Netezza::SQLAPI *"parameter 1", "parameter n"*

Netezza::

is required to distinguish special queries from regular queries.

SQLAPI

is the specific API that is being called. Both Netezza:: and SQLAPI are case sensitive.

*"parameter n"*

is a quoted string that is delimited by commas.

Within the quoted string, two characters are universally recognized: the percent sign (%) and the underscore (\_). The percent sign matches any sequence of zero or more characters, and the underscore represents any single character. To use either character as a literal value, you can use the backslash character (\) to escape the match characters. For example, the following call to SQLTables usually matches table names such as myatest and my\_test:

```
select * from connection to netezza (NETEZZA::SQLTables "test","", "my_test");
```

Use the escape character to search only for the table, my\_test:

```
select * from connection to netezza (NETEZZA::SQLTables "test","", "my\_test");
```

SAS/ACCESS Interface to Netezza supports these special queries:

Netezza::SQLTables <"Catalog", "Schema", "Table-name", "Type">

returns a list of all tables that match the specified arguments. If you do not specify any arguments, all accessible table names and information are returned.

Netezza::SQLColumns <"Catalog", "Schema", "Table-name", "Column-name">

returns a list of all columns that match the specified arguments. If you do not specify any argument, all accessible column names and information are returned.

Netezza::SQLPrimaryKeys <"Catalog", "Schema", "Table-name">

returns a list of all columns that compose the primary key that matches the specified table. A primary key can be composed of one or more columns. If you do not specify any table name, this special query fails.

Netezza::SQLSpecialColumns <"Identifier-type", "Catalog-name", "Schema-name", "Table-name", "Scope", "Nullable">

returns a list of the optimal set of columns that uniquely identify a row in the specified table.

Netezza::SQLStatistics <"Catalog", "Schema", "Table-name">  
 returns a list of the statistics for the specified table name, with options of SQL\_INDEX\_ALL and SQL\_ENSURE set in the SQLStatistics API call. If you do not specify any table name argument, this special query fails.

Netezza::SQLGetTypeInfo  
 returns information about the data types that the Netezza Performance Server supports.

---

## Passing SAS Functions to Netezza

SAS/ACCESS Interface to Netezza passes the following SAS functions to the data source for processing. Where the Netezza function name differs from the SAS function name, the Netezza name appears in parentheses. See the section about optimizing SQL usage in *SAS/ACCESS for Relational Databases: Reference* for information.

ABS  
 ARCOS (ACOS)  
 ARSIN (ASIN)  
 ATAN  
 ATAN2  
 AVG  
 BAND (int4and)  
 BNOT (int4not)  
 BLSHIFT (int4shl)  
 BRSHIFT (int4shr)  
 BOR (int4or)  
 BXOR (int4xor)  
 BYTE (chr)  
 CEIL  
 COALESCE  
 COMPRESS (translate)  
 COS  
 COUNT  
 DAY (date\_part)  
 EXP  
 FLOOR  
 HOUR (date\_part)  
 INDEX (position)  
 LOG (ln)  
 LOG10 (log)  
 LOWCASE (lower)  
 MAX  
 MIN  
 MINUTE (date\_part)  
 MOD  
 MONTH (date\_part)

REPEAT  
 SECOND (date\_part)  
 SIGN  
 SIN  
 SOUNDEX  
 SQRT  
 STRIP (btrim)  
 SUBSTR  
 SUM  
 TAN  
 TRANWRD (translate)  
 TRIMN (rtrim)  
 UPCASE (upper)  
 YEAR (date\_part)

If SQL\_FUNCTIONS=ALL, these options are passed down:

DATE (current\_date)  
 DATEPART (cast)  
 DATETIME (now)  
 LENGTH  
 ROUND  
 TIME (current\_time)  
 TIMEPART (cast)  
 TODAY (current\_date)  
 TRANSLATE

---

## Passing Joins to Netezza

For a multiple libref join to pass to Netezza, all of these components of the LIBNAME statements must match exactly:

user ID (USER=)  
 password (PASSWORD=)  
 server (SERVER=)  
 database (DATABASE=)  
 port (PORT=)  
 datasource (DSN=, if specified)  
 catalog (QUALIFIER=, if specified)  
 SQL functions (SQL\_FUNCTIONS=)

See the section about performance considerations in *SAS/ACCESS for Relational Databases: Reference* for more information about when and how SAS/ACCESS Interface to Netezza passes joins to the DBMS.

---

# Temporary Table Support for Netezza

---

## General Information

See the section on the temporary table support in *SAS/ACCESS for Relational Databases: Reference* for general information about this feature.

---

## Establishing a Temporary Table

To make full use of temporary tables, the CONNECTION=GLOBAL connection option is necessary. This option lets you use a single connection across SAS DATA steps and SAS procedure boundaries. This connection can also be shared between LIBNAME statements and the Pass-Through Facility. Because a temporary table exists only within a single connection, you need to be able to share this single connection among all steps that reference the temporary table. The temporary table cannot be referenced from any other connection.

You can currently use only a PROC SQL statement to create a temporary table. To use both the Pass-Through Facility and librefs to reference a temporary table, you must specify a LIBNAME statement before the PROC SQL step so that global connection persists across SAS steps and even across multiple PROC SQL steps. Here is an example:

```
proc sql;
  connect to netezza (server=nps1 database=test
    user=myuser password=mypwd connection=global);
  execute (create temporary table temptabl as select * from permtable ) by netezza;
quit;
```

At this point, you can refer to the temporary table by using either the Temp libref or the CONNECTION=GLOBAL option with a PROC SQL step.

---

## Terminating a Temporary Table

You can drop a temporary table at any time or allow it to be implicitly dropped when the connection is terminated. Temporary tables do not persist beyond the scope of a single connection.

---

## Examples

The following assumptions apply to the examples in this section:

- The DeptInfo table already exists on the DBMS that contains all of your department information.
- One SAS data set contains join criteria that you want to use to extract specific rows from the DeptInfo table.
- Another SAS data set contains updates to the DeptInfo table.

These examples use the following librefs and temporary tables.

```
libname saslib base 'SAS-Data-Library';
libname dept netezza server=npsl database=test user=myuser pwd=mypwd connection=global;
libname temp netezza server=npsl database=test user=myuser pwd=mypwd connection=global;

proc sql;
    connect to netezza (server=npsl database=test user=myuser pwd=mypwd
                        connection=global);
    execute (create temporary table temptabl (dname char(20), deptno int))
            by netezza;
quit;
```

This first example shows how to use a heterogeneous join with a temporary table to perform a homogeneous join on the DBMS, instead of reading the DBMS table into SAS to perform the join. By using the table that was created previously, you can copy SAS data into the temporary table to perform the join.

```
proc sql;
    connect to netezza (server=npsl database=test user=myuser pwd=mypwd
                        connection=global);
    insert into temp.temptabl select * from saslib.joindata;
    select * from dept.deptinfo info, temp.temptabl tab
        where info.deptno = tab.deptno;
    /* remove the rows for the next example */
    execute (delete from temptabl) by netezza;
quit;
```

In this next example, transaction processing on the DBMS occurs by using a temporary table instead of using either DBKEY= or MULTI\_DATASRC\_OPT=IN\_CLAUSE with a SAS data set as the transaction table.

```
proc sql;
    connect to netezza (server=npsl database=test user=myuser pwd=mypwd
                        connection=global);
    insert into temp.temptabl select * from saslib.transdat;
    execute (update deptinfo d set dname = (select dname from temptabl)
            where d.deptno = (select deptno from temptabl)) by netezza;
quit;
```

---

## Netezza Bulk Loading and Unloading

---

### Loading

Bulk loading is the fastest way to insert large numbers of rows into a Netezza table. To use the bulk-load facility, specify BULKLOAD=YES. The bulk-load facility uses the Netezza Remote External Table interface to move data from the client to the Netezza Performance Server.

Here are the Netezza bulk-load data set options:

```
BL_DATAFILE=
BL_DATAFILE=
BL_DELIMITER=
```

```
BL_OPTIONS=
BL_USE_PIPE=
BULKLOAD=
```

## Examples

This first example shows how you can use a SAS data set, SASFLT.FLT98, to create and load a large Netezza table, FLIGHTS98:

```
libname sasflt 'SAS-data-library';
libname net_air netezza user=louis pwd=fromage
      server=air2 database=flights;

proc sql;
create table net_air.flights98
      (bulkload=YES bl_options='logdir "c:\temp\netlogs"')
      as select * from sasflt.flt98;
quit;
```

You can use BL\_OPTIONS= to pass Netezza-specific options to the bulk-loading process. The logdir option specifies the directory for the nzbad and nzlog files to be generated during the load.

This next example shows how you can append the SAS data set, SASFLT.FLT98, to the existing Netezza table, ALLFLIGHTS. The BL\_USE\_PIPE=NO option forces SAS/ACCESS Interface to Netezza to write data to a flat file, as specified in the BL\_DATAFILE= option. Rather than deleting the data file, BL\_DELETE\_DATAFILE=NO causes the engine to leave it after the load has completed.

```
proc append base=net_air.allflights
      (BULKLOAD=YES
      BL_DATAFILE='/tmp/fltdata.dat'
      BL_USE_PIPE=NO
      BL_DELETE_DATAFILE=NO)
data=sasflt.flt98;
run;
```

---

## Unloading

Bulk unloading is the fastest way to insert large numbers of rows from a Netezza table. To use the bulk-unload facility, specify BULKUNLOAD=YES. The bulk-unload facility uses the Netezza Remote External Table interface to move data from the client to the Netezza Performance Server into SAS.

Here are the Netezza bulk-unload data set options:

```
BL_DATAFILE=
BL_DATAFILE=
BL_DELIMITER=
BL_OPTIONS=
BL_USE_PIPE=
BULKLOAD=
```



## Examples

This first example shows how you can read the large Netezza table, `FLIGHTS98`, to create and populate a SAS data set, `SASFLT.FLT98`:

```
libname sasflt 'SAS-data-library';
libname net_air netezza user=louis pwd=fromage
        server=air2 database=flights;

proc sql;
create table sasflt.flt98
    as select * from net_air.flights98
    (bulkunload=YES bl_options='logdir "c:\temp\netlogs"');
quit;
```

You can use `BL_OPTIONS=` to pass Netezza-specific options to the unload process. The `logdir` option specifies the directory for the `nzbad` and `nzlog` files to be generated during the unload.

This next example shows how you can append the contents of the Netezza table, `ALLFLIGHTS`, to an existing SAS data set, `SASFLT.FLT98`. The `BL_USE_PIPE=NO` option forces SAS/ACCESS Interface to Netezza to read data from a flat file, as specified in the `BL_DATAFILE=` option. Rather than deleting the data file, `BL_DELETE_DATAFILE=NO` causes the engine to leave it after the unload has completed.

```
proc append base=sasflt.flt98
data=net_air.allflights
    (BULKUNLOAD=YES
    BL_DATAFILE='/tmp/fltdata.dat'
    BL_USE_PIPE=NO
    BL_DELETE_DATAFILE=NO);
run;
```

---

## Naming Conventions for Netezza

Beginning in SAS Version 7, most SAS names can be up to 32 characters long. SAS/ACCESS Interface to Netezza supports table names and column names that contain up to 32 characters. If DBMS column names are longer than 32 characters, they are truncated to 32 characters. If truncating a column name would result in identical names, SAS generates a unique name by replacing the last character with a number. DBMS table names must be 32 characters or less because SAS does not truncate a longer name. If you already have a table name that is greater than 32 characters, it is recommended that you create a table view.

The `PRESERVE_COL_NAMES=` and `PRESERVE_TAB_NAMES=` options determine how SAS/ACCESS Interface to Netezza handles case sensitivity. Netezza is not case sensitive, and all names default to lowercase.

Netezza objects include tables, views, and columns. Follow these naming conventions:

- ☐ A name must be from 1 to 128 characters long.
- ☐ A name must begin with a letter (A through Z), diacritic marks, or non-Latin characters (200-377 octal).
- ☐ A name cannot begin with an underscore (`_`). Leading underscores are reserved for system objects.

- Names are not case sensitive. For example, **CUSTOMER** and **Customer** are the same, but object names are converted to lowercase when they are stored in the Netezza database. However, if you enclose a name in quotation marks, it is case sensitive.
- A name cannot be a Netezza reserved word, such as WHERE or VIEW.
- A name cannot be the same as another Netezza object that has the same type.

For more information, see your *Netezza Database User's Guide*.

---

## Data Types for Netezza

---

### Overview

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

This information includes information about Netezza data types, null and default values, and data conversions. For more information about Netezza data types and to determine which data types are available for your version of Netezza, see your *Netezza Database User's Guide*.

*Note:* SAS/ACCESS Interface to Netezza does not directly support NCHAR, NVARCHAR, TIMETZ, or INTERVAL types. Any columns using these types are read into SAS as character strings. △

---

### String Data

#### CHAR(*n*)

specifies a fixed-length column for character string data. The maximum length is 32,768 characters.

#### VARCHAR(*n*)

specifies a varying-length column for character string data. The maximum length is 32,768 characters.

---

### Numeric Data

#### BIGINT

specifies a big integer. Values in a column of this type can range from -9223372036854775808 to +9223372036854775807.

#### SMALLINT

specifies a small integer. Values in a column of this type can range from -32768 through +32767.

#### INTEGER

specifies a large integer. Values in a column of this type can range from -2147483648 through +2147483647.

**BYTEINT**

specifies a tiny integer. Values in a column of this type can range from –128 to +127.

**DOUBLE | DOUBLE PRECISION**

specifies a floating-point number that is 64 bits long. Values in a column of this type can range from –1.79769E+308 to –2.225E-307 or +2.225E-307 to +1.79769E+308, or they can be 0. This data type is stored the same way that SAS stores its numeric data type. Therefore, numeric columns of this type require the least processing when SAS accesses them.

**REAL**

specifies a floating-point number that is 32 bits long. Values in a column of this type can range from approximately –3.4E38 to –1.17E-38 and +1.17E-38 to +3.4E38.

**DECIMAL | DEC | NUMERIC | NUM**

specifies a fixed-point decimal number. The precision and scale of the number determines the position of the decimal point. The numbers to the right of the decimal point are the scale, which cannot be negative or greater than the precision. The maximum precision is 38 digits.

## Dates, Times, and Timestamps

SQL date and time data types are collectively called datetime values. The SQL data types for dates, times, and timestamps are listed here. Be aware that columns of these data types can contain data values that are out of range for SAS.

**DATE**

specifies date values. The range is 01-01-0001 to 12-31-9999. The default format *YYYY-MM-DD*—for example, 1961–06–13. Netezza supports many other formats for entering date data. For more information, see your *Netezza Database User's Guide*.

**TIME**

specifies time values in hours, minutes, and seconds to six decimal positions: *hh:mm:ss[.nnnnnn]*. The range is 00:00:00.000000 to 23:59:59.999999. However, due to the ODBC-style interface that SAS/ACCESS Interface to Netezza uses to communicate with the Netezza Performance Server, any fractional seconds are lost in the transfer of data from server to client.

**TIMESTAMP**

combines a date and time in the default format of *yyyy-mm-dd hh:mm:ss[.nnnnnn]*. For example, a timestamp for precisely 2:25 p.m. on January 25, 1991, would be 1991-01-25-14.25.00.000000. Values in a column of this type have the same ranges as described for DATE and TIME.

## Netezza Null Values

Netezza has a special value called NULL. A Netezza NULL value means an absence of information and is analogous to a SAS missing value. When SAS/ACCESS reads a Netezza NULL value, it interprets it as a SAS missing value.

You can define a column in a Netezza table so that it requires data. To do this in SQL, you specify a column as NOT NULL, which tells SQL to allow only a row to be added to a table if a value exists for the field. For example, NOT NULL assigned to the CUSTOMER field in the SASDEMO.CUSTOMER table does not allow a row to be

added unless there is a value for CUSTOMER. When creating a Netezza table with SAS/ACCESS, you can use the DBNULL= data set option to indicate whether NULL is a valid value for specified columns.

You can also define Netezza columns as NOT NULL DEFAULT. For more information about using the NOT NULL DEFAULT value, see your *Netezza Database User's Guide*.

Knowing whether a Netezza column allows NULLs or whether the host system supplies a default value for a column that is defined as NOT NULL DEFAULT can help you write selection criteria and enter values to update a table. Unless a column is defined as NOT NULL or NOT NULL DEFAULT, it allows NULL values.

For more information about how SAS handles NULL values, see “Potential Result Set Differences When Processing Null Data” in *SAS/ACCESS for Relational Databases: Reference*.

To control how SAS missing character values are handled by the DBMS, use the NULLCHAR= and NULLCHARVAL= data set options.

---

## LIBNAME Statement Data Conversions

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

**Table 1.3** LIBNAME Statement: Default SAS Formats for Netezza Data Types

Netezza Data Type	SAS Data Type	Default SAS Format
CHAR( <i>n</i> )	character	<i>\$n.</i>
VARCHAR( <i>n</i> )	character	<i>\$n.</i>
INTEGER	numeric	11.
SMALLINT	numeric	6.
BYTEINT	numeric	4.
BIGINT	numeric	20.
DECIMAL	numeric	<i>m.n</i>
NUMERIC	numeric	<i>m.n</i>
REAL	numeric	none
DOUBLE	numeric	none
TIME	numeric	TIME8.
DATE	numeric	DATE9.
TIMESTAMP	numeric	DATETIME25.6

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

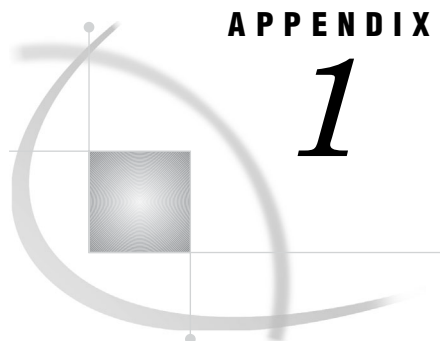
The following table shows the default Netezza data types that SAS/ACCESS assigns to SAS variable formats during output operations when you use the LIBNAME statement.

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

SAS Variable Format	Netezza Data Type
<i>m.n</i>	DECIMAL ( <i>m,n</i> )
other numerics	DOUBLE
<i>\$n.</i>	VARCHAR( <i>n</i> )
datetime formats	TIMESTAMP
date formats	DATE
time formats	TIME

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





## APPENDIX

## 1

## Recommended Reading

---

*Recommended Reading* 19

---

### Recommended Reading

Here is the recommended reading list for this title:

- ☐ *SAS/ACCESS for Relational Databases: Reference*
- ☐ *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](mailto:sasbook@sas.com)

Web address: [support.sas.com/pubs](http://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

---

## B

- BIGINT data type
  - Netezza 14
- bulk loading
  - Netezza 11
- bulk unloading
  - Netezza 12
- BYTEINT data type
  - Netezza 15

## C

- CHAR data type
  - Netezza 14
- CONNECT statement, SQL procedure
  - Netezza 6

## D

- data conversions
  - Netezza 16
- data set options
  - Netezza 5
- data types
  - Netezza 14
- DATABASE= option, LIBNAME statement
  - Netezza 2
- DATE data type
  - Netezza 15
- datetime values
  - Netezza 15
- DEC data type
  - Netezza 15
- DECIMAL data type
  - Netezza 15
- DOUBLE data type
  - Netezza 15
- DOUBLE PRECISION data type
  - Netezza 15
- DSN= option, LIBNAME statement
  - Netezza 3

## F

- functions
  - passing to Netezza 8

## I

- INTEGER data type
  - Netezza 14

## J

- joins
  - passing to Netezza 9

## L

- LIBNAME statement
  - Netezza 2

## N

- naming conventions
  - Netezza 13
- Netezza 1
  - bulk loading and unloading 11
  - data set options 5
  - data types 14
  - LIBNAME statement 2
  - naming conventions 13
  - Pass-Through Facility 6
  - passing functions to 8
  - passing joins to 9
  - special catalog queries 7
  - temporary table support 10
- NULL values
  - Netezza 15
- NUM data type
  - Netezza 15
- numeric data
  - Netezza 14
- NUMERIC data type
  - Netezza 15

## P

- Pass-Through Facility
  - Netezza 6
- PASSWORD= option, LIBNAME statement
  - Netezza 3
- PORT= option, LIBNAME statement
  - Netezza 2

**Q**

queries  
    Netezza special catalog queries 7

**R**

READ\_ONLY= option, LIBNAME statement  
    Netezza 3  
REAL data type  
    Netezza 15

**S**

SERVER= option, LIBNAME statement  
    Netezza 2  
SMALLINT data type  
    Netezza 14  
string data  
    Netezza 14

**T**

temporary tables  
    Netezza 10  
TIME data type  
    Netezza 15  
TIMESTAMP data type  
    Netezza 15

**U**

USER= option, LIBNAME statement  
    Netezza 2

**V**

VARCHAR data type  
    Netezza 14



# Your Turn

---

We want your feedback.

- ☐ If you have comments about this book, please send them to **`yourturn@sas.com`**. Include the full title and page numbers (if applicable).
- ☐ If you have comments about the software, please send them to **`suggest@sas.com`**.



# SAS® Publishing gives you the tools to flourish in any environment with SAS!

Whether you are new to the workforce or an experienced professional, you need to distinguish yourself in this rapidly changing and competitive job market. SAS® Publishing provides you with a wide range of resources to help you set yourself apart.

## **SAS® Press Series**

Need to learn the basics? Struggling with a programming problem? You'll find the expert answers that you need in example-rich books from the SAS Press Series. Written by experienced SAS professionals from around the world, these books deliver real-world insights on a broad range of topics for all skill levels.

**[support.sas.com/saspress](http://support.sas.com/saspress)**

## **SAS® Documentation**

To successfully implement applications using SAS software, companies in every industry and on every continent all turn to the one source for accurate, timely, and reliable information—SAS documentation. We currently produce the following types of reference documentation: online help that is built into the software, tutorials that are integrated into the product, reference documentation delivered in HTML and PDF—free on the Web, and hard-copy books.

**[support.sas.com/publishing](http://support.sas.com/publishing)**

## **SAS® Learning Edition 4.1**

Get a workplace advantage, perform analytics in less time, and prepare for the SAS Base Programming exam and SAS Advanced Programming exam with SAS® Learning Edition 4.1. This inexpensive, intuitive personal learning version of SAS includes Base SAS® 9.1.3, SAS/STAT®, SAS/GRAPH®, SAS/QC®, SAS/ETS®, and SAS® Enterprise Guide® 4.1. Whether you are a professor, student, or business professional, this is a great way to learn SAS.

**[support.sas.com/LE](http://support.sas.com/LE)**



**THE  
POWER  
TO KNOW®**



























