Introduction

This chapter introduces SAS System users to ORACLE, a relational database management system (DBMS). It accompanies and should be used with SAS/ACCESS Software for Relational Databases: Reference, First Edition (order #55940).*

This chapter focuses on the terms and concepts that help you use the SAS/ACCESS Interface to ORACLE. Then it describes the statements and options that are specific to ORACLE, the ACCESS and DBLOAD procedures, and the SQL procedure's CONNECT statement.

For more information on an ORACLE concept or term, refer to your ORACLE documentation.

* Copyright © 1999 by SAS Institute Inc., Cary, NC, USA. All rights reserved.
Version 7 Information

“SAS/ACCESS LIBNAME Statement” on page 24 describes options that you specify in the LIBNAME statement to associate a SAS libref with a DBMS database, schema, server, or group of tables and views. The following section describes DBMS-specific options and option values that you can use in the SAS/ACCESS Interface to ORACLE.

Dictionary

LIBNAME Statement: ORACLE Specifics

Associates a SAS libref with a DBMS database, schema, server, or group of tables and views.

Valid: Anywhere

Syntax

LIBNAME libref SAS/ACCESS-engine-name
   <SAS/ACCESS-engine-connection-options> <SAS/ACCESS-LIBNAME-options>;

Arguments

libref

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

SAS/ACCESS-engine-name

is the SAS/ACCESS engine name for your DBMS, in this case, oracle

SAS/ACCESS engines are implemented differently in different operating environments. The engine name is required.

SAS/ACCESS-engine-connection-options

are options that you specify to connect to ORACLE. If the connection options contain characters that are not allowed in SAS names, enclose the values of the options in quotation marks. If you specify the appropriate system options or environment variables for ORACLE, you can often omit the SAS/ACCESS engine connection options. See your ORACLE documentation for details.

SAS/ACCESS-LIBNAME-options

are options that apply to the processing of objects and data in a DBMS, such as its tables or indexes. For example, the UPDATEBUFF= option enables you to specify the number of rows to update or delete in a single ORACLE UPDATE or DELETE transaction. Support for many of these options is DBMS specific.
Some SAS/ACCESS LIBNAME options have the same names as SAS/ACCESS data set options. When you specify an option in the LIBNAME statement, it applies to objects and data that are referenced by the libref. A SAS/ACCESS data set option applies only to the data set on which it is specified. If a like-named option is specified in both the SAS/ACCESS engine LIBNAME statement and after a data set name (which references a DBMS table or view), the SAS System uses the value that is specified later, on the data set name. See “Data Set Options: ORACLE Specifics” on page 380 for more information.

Details

The LIBNAME statement associates a libref with a SAS/ACCESS engine to access tables or views in a database management system (DBMS). The SAS/ACCESS engine enables you to connect to a particular DBMS and to specify a DBMS table or view name in a two-level SAS name.

For example, in MYDBLIB.EMPLOYEES_Q2, MYDBLIB is a SAS libref that points to a particular group of DBMS objects, and EMPLOYEES_Q2 is a DBMS table name. When you specify MYDBLIB.EMPLOYEES_Q2 in a DATA step or procedure, you dynamically access the DBMS table. Version 7 of the SAS System supports reading, updating, creating, and deleting DBMS tables dynamically.

To disassociate or clear a libref from a DBMS, use a LIBNAME statement, specifying the libref (for example, MYDBLIB) and the CLEAR options as follows:

```
libname mydblib CLEAR;
```

The database engine will disconnect from the database and close any free threads or resources that are associated with that connection.

SAS/ACCESS Engine Connection Options

The SAS/ACCESS engine connection options for ORACLE are as follows:

```
USER= on page 375
PASSWORD= on page 375
PATH= on page 375
USER=<>ORACLE-user-name<>
    specifies an optional ORACLE user name. If the user name contains blanks or national characters, enclose the name in quotation marks. If you omit an ORACLE user name and password, the default ORACLE user ID OPS$sysid is used, if it is enabled. USER= must be used with PASSWORD=

    USER= can also be specified with the USERNAME= alias.

PASSWORD=<>ORACLE-password<>
    specifies an optional ORACLE password that is associated with the ORACLE user name. If you omit PASSWORD=, the password for the default ORACLE user ID OPS$sysid is used, if it is enabled. PASSWORD= must be used with USER=

    PASSWORD= can also be specified with the PASS=, PW=, ORACLEPW=, and ORAPW= aliases.

PATH=<>ORACLE-database-specification<>
    specifies the ORACLE driver, node, and database. Aliases are required if you are using SQL*Net Version 2.0 or later. In some operating environments, you can enter the information that is required by the PATH= statement before invoking the SAS System.
```
SAS/ACCESS uses the same ORACLE path designation that you use to connect to ORACLE directly. See your database administrator to determine the databases that have been set up in your operating environment, and to determine the default values if you do not specify a database. On UNIX systems, the TWO_TASK environment variable is used, if set. If neither PATH= nor TWO_TASK have been set, the default value is the local driver.

**SAS/ACCESS LIBNAME Options**

When you specify any of the following options in the LIBNAME statement, the option is applied to all objects (such as tables and views) in the database that the libref represents.

The SAS/ACCESS LIBNAME options for ORACLE are as follows:

- **DBINDEX=** on page 376
- **DBLINK=** on page 376
- **DBPROMPT=** on page 376
- **INSERTBUFF=** on page 377
- **LOCKWAIT=** on page 377
- **ORACLE_73_OR_ABOVE=** on page 377
- **PRESERVE_COL_NAMES=** on page 378
- **PRESERVE_TAB_NAMES=** on page 378
- **READ_ISOLATION_LEVEL=** on page 378
- **READ_LOCK_TYPE=** on page 378
- **READBUFF=** on page 379
- **SCHEMA=** on page 379
- **SPOOL=** on page 379
- **UPDATE_ISOLATION_LEVEL=** on page 379
- **UPDATE_LOCK_TYPE=** on page 379
- **UPDATEBUFF=** on page 380

**DBINDEX=**YES | NO

indicates whether SAS calls the DBMS to find indexes on the specified table.

If you omit DBINDEX=, the default value is DBINDEX=NO.

For a full description of this option, refer to .

**DBLINK=**database-link

specifies a link in the local database that enables access to tables and views in a remote database.

If you omit DBLINK=, SAS accesses objects in the local database.

A link is a database object that is used to identify an object stored in a remote database. A link contains stored path information and may also contain user name and password information for connecting to the remote database.

**DBPROMPT=**YES | NO

specifies whether SAS displays a prompting window that enables you to enter SAS/ACCESS engine connection options instead of specifying them on the LIBNAME statement.
If you omit DBPROMPT=, the default value is DBPROMPT=NO. The SAS/ACCESS Interface to ORACLE allows you to enter 30 characters for the USERNAME and PASSWORD and up to 70 characters for the PATH, depending on your platform.

For a full description of this option, refer to .

**INSERTBUFF=positive-integer**

specifies the number of rows in a single ORACLE insert operation.

If you omit INSERTBUFF=, the default value is INSERTBUFF=10. SAS allows the maximum number that is allowed by ORACLE.

Note: When you assign a value that is greater than INSERTBUFF=1, the SAS application notes that indicate the success or failure of the insert operation may be incorrect because these notes only represent information for a single insert, even when multiple inserts are performed. △

Note: If specified, the value of the DBCOMMIT= option overrides the value of INSERTBUFF= △

**LOCKWAIT=YES | NO**

specifies whether to wait indefinitely until rows are available for locking.

If you specify LOCKWAIT=YES, SAS waits until rows are available for locking. If you specify LOCKWAIT=NO, SAS does not wait and returns an error to indicate that the lock is not available. If you omit LOCKWAIT=, the default value is LOCKWAIT=YES.

**ORACLE_73_OR_ABOVE=YES | NO**

specifies whether the ORACLE server version is 7.3 or later.

If you specify ORACLE_73_OR_ABOVE=YES or omit this option, SAS can use the SERIALIZABLE isolation level for update locking that is available in ORACLE 7.3 and above. Users with version 7.3 or above may set the ORACLE_73_OR_ABOVE= option to either YES or NO.

For Oracle versions prior to 7.3, updates without locking are performed as they were in SAS Version 6. In Version 6, a row is updated with an additional WHERE clause to ensure that the row has not changed since the time it was read. The update fails if the row has changed. For versions 7.3 and above, updates are performed in serializable transactions. An update on a row automatically fails if the row has been changed since the time the serializable transaction started. (This is not always true; due to current, published ORACLE bug 440366, sometimes an update on a row fails even if the row has not changed. ORACLE offers the following solution: When creating a table, users can increase the number of INITRANS to at least 3 for the table.)

In a scenario where ORACLE_73_OR_ABOVE= is incorrectly set to YES when it should be NO, the Oracle engine detects this error and automatically makes the assumption that the Oracle version is below 7.3. In a scenario where ORACLE_73_OR_ABOVE= is incorrectly set to NO when it should be YES, the Oracle engine does not detect the incorrect setting. The update is performed without using a serializable transaction.

The advantages of setting ORACLE_73_OR_ABOVE=YES are that no extra WHERE clause overhead is incurred, and WHERE clause floating point number comparison problems (precision problems) are avoided.

ORACLE_73_OR_ABOVE= can also be specified with the ORACLE_73= alias.
LIBNAME Statement: ORACLE Specifics

Chapter 20

See also READ_ISOLATION_LEVEL=, UPDATE_ISOLATION_LEVEL=,

PRESERVE_COL_NAMES=YES | NO

preserves spaces, special characters, and mixed case in column names.

If you omit PRESERVE_COL_NAMES=, the default value is
PRESERVE_COL_NAMES=NO. If you want to preserve the case or allow
characters that are not supported in SAS names, such as '$', in your column
names, set PRESERVE_COL_NAMES=YES.

For a full description of this option, refer to.

PRESERVE_TAB_NAMES=YES | NO

preserves spaces, special characters, and mixed case in table names.

If you omit PRESERVE_TAB_NAMES=, the default value is
PRESERVE_TAB_NAMES=NO. If you want to preserve case or allow characters
that are not supported in SAS names, such as '$', in your object names, including
table names, schema names, and link names, set PRESERVE_TAB_NAMES=YES.

For a full description of this option, refer to.

READ_ISOLATION_LEVEL=READCOMMITTED | SERIALIZABLE

specifies which isolation level for ORACLE to use when it reads tables and views.

If you are using ORACLE Version 7.3 or later, you can set
READ_ISOLATION_LEVEL= to READCOMMITTED or SERIALIZABLE. If you
specify READ_ISOLATION_LEVEL=READCOMMITTED, SAS uses the standard
method of read locking that is available in all ORACLE versions. If you specify
READ_ISOLATION_LEVEL=SERIALIZABLE, SAS uses the SERIALIZABLE
method of read locking that is available in ORACLE versions 7.3 and later. The
ORACLE_73= option must be set to ORACLE_73=YES to use the SERIALIZABLE
isolation level.

The SPOOL= option overrides the READ_ISOLATION_LEVEL= option. If
SPOOL=DBMS, SAS automatically sets
READ_ISOLATION_LEVEL=SERIALIZABLE for ORACLE 7.3 or later and
READ_ONLY for prior versions. If you omit READ_ISOLATION_LEVEL=, the
default value is based on the value of the SPOOL= option. In all other cases, the
default value is READCOMMITTED.

When READ_ISOLATION_LEVEL is set to SERIALIZABLE, the
CONNECTION option must be set to UNIQUE. If not, an error occurs.

Note: This option should be rarely needed because the SAS/ACCESS engine
chooses the appropriate isolation level based on other locking options.

See also UPDATE_ISOLATION_LEVEL=,

READ_LOCK_TYPE=NOLOCK | ROW | TABLE

specifies how a table is locked during read operations.

If you omit READ_LOCK_TYPE=, the default value is
READ_LOCK_TYPE=NOLOCK. If you specify READ_LOCK_TYPE=NOLOCK,
table locking is not used during the reading of tables and views. If you specify
READ_LOCK_TYPE=ROW, the ORACLE "ROW SHARE" table lock is used during
the reading of tables and views. If you specify READ_LOCK_TYPE=TABLE, the
ORACLE "SHARE" table lock is used during the reading of tables and views.

When READ_LOCK_TYPE is set to either TABLE or ROW, the CONNECTION
option must be set to UNIQUE. If not, an error occurs.
For a full description of this option, refer to

**READBUFF** = positive-integer

specifies the number of rows in a single ORACLE fetch.

If you omit **READBUFF**, the default value is **READBUFF** = 25. SAS allows the maximum number that is allowed by ORACLE.

**READBUFF** can also be specified with the **BUFFSIZE** = alias.

**SCHEMA** = schema-name

specifies a schema name to be used when referring to database objects. SAS can access another user's database objects by using a specified schema name.

If you omit **SCHEMA**, SAS accesses objects in the default and public schemas. If **PRESERVE_TAB_NAMES** = NO, SAS converts the **SCHEMA** value to uppercase because all values in the ORACLE data dictionary are uppercase unless quoted.

**SPOOL** = **YES** | **NO** | **DBMS**

specifies whether SAS creates a utility spool file during read operations that are performed with the specified libref.

If you omit **SPOOL**, the default value is **SPOOL** = **YES**. If **SPOOL** = **DBMS**, the SAS/ACCESS Interface to ORACLE satisfies the two-pass requirement by starting a read-only transaction. **SPOOL** = **YES** and **SPOOL** = **DBMS** have comparable performance results for ORACLE.

When **SPOOL** is set to **DBMS**, the **CONNECTION** option must be set to **UNIQUE**. If not, an error occurs.

For a full description of this option, refer to.

**UPDATE_ISOLATION_LEVEL** = **READCOMMITTED** | **SERIALIZABLE**

specifies which isolation level for ORACLE to use when it updates tables and views.

If you are using ORACLE Version 7.3 or later, you can set **UPDATE_ISOLATION_LEVEL** to **READCOMMITTED** or **SERIALIZABLE**. If you specify **UPDATE_ISOLATION_LEVEL** = **READCOMMITTED**, SAS uses the standard method of update locking that is available in all ORACLE versions. If you specify **UPDATE_ISOLATION_LEVEL** = **SERIALIZABLE**, SAS uses the **SERIALIZABLE** method of update locking that is available in ORACLE versions 7.3 and later. The **ORACLE_73** option must be set to **ORACLE_73** = **YES** to use the **SERIALIZABLE** isolation level.

If you omit **UPDATE_ISOLATION_LEVEL**, the default value is based on the value of the **UPDATE_LOCK_TYPE** = option. If **UPDATE_LOCK_TYPE** = **NOLOCK**, SAS automatically sets **UPDATE_ISOLATION_LEVEL** = **SERIALIZABLE** for ORACLE 7.3 or later and **READONLY** for prior versions. In all other cases, the default value is **READCOMMITTED**.

Note: This option should be rarely needed because the SAS/ACCESS engine chooses the appropriate isolation level based on other locking options.

**UPDATE_LOCK_TYPE** = **NOLOCK** | **ROW** | **TABLE**

specifies how a table is locked during update operations.

If you omit **UPDATE_LOCK_TYPE**, the default value is **UPDATE_LOCK_TYPE** = **NOLOCK**. If you specify **UPDATE_LOCK_TYPE** = **NOLOCK**, table locking is not used during the reading of tables and views for update. If you specify **UPDATE_LOCK_TYPE** = **ROW**, the
ORACLE “ROW SHARE” table lock is used during the reading of tables and views for update. If you specify UPDATE_LOCK_TYPE=TABLE, the ORACLE “EXCLUSIVE” table lock is used during the reading of tables and views for update.

If UPDATE_LOCK_TYPE=NOLOCK and ORACLE_73=YES, updates are performed using serializable transactions. If UPDATE_LOCK_TYPE=NOLOCK and ORACLE_73=NO, updates are performed using an extra WHERE clause to ensure that the row has not been updated since it was first read. Updates might fail when UPDATE_LOCK_TYPE=NOLOCK because other users might modify a row after the row was read for update.

If the ORACLE_73_OR_ABOVE= option is incorrectly set to YES (meaning that the Oracle server version is below 7.3), the Oracle engine detects this, and the update is performed as if ORACLE_73_OR_ABOVE= were correctly set.

For a full description of this option, refer to.

`UPDATEBUFF=positive-integer`

specifies the number of rows in a single ORACLE update/delete operation.

If you omit `UPDATEBUFF=`, the default value is `UPDATEBUFF=1`. SAS allows the maximum that ORACLE allows.

Example: Specifying a LIBNAME Statement to Access ORACLE Data

In this example, the libref MYDBLIB uses the SAS/ACCESS Interface to ORACLE to connect to an ORACLE database. The SAS/ACCESS engine connection options are USER=, PASSWORD=, and PATH=, where PATH= specifies an alias for the database specification.

```
libname mydblib oracle user=scott
  password=tiger path='myorapath';
```

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

Data Set Options: ORACLE Specifics

"SAS/ACCESS Data Set Options" on page 39 describes the SAS/ACCESS options that you can use when you specify a SAS data set in a DATA or PROC step; in this case, the SAS data set accesses data from a DBMS table or view. The following section describes the DBMS-specific options and option values that you can use in the SAS/ACCESS Interface to ORACLE. A data set option applies only to the data set, or DBMS object, on which it is specified.

Unless otherwise noted, when you omit a SAS/ACCESS data set option, and you have specified a like-named LIBNAME option in the LIBNAME statement, the value of the LIBNAME option applies to all data sets within the libref.

Note: Not all LIBNAME options have corresponding data set options.

The SAS/ACCESS data set options for ORACLE are as follows:

- "DBFORCE=" on page 381
- "DBINDEX=" on page 381
- "DBLINK=" on page 381
**DBFORCE=**

Specifies whether to force the truncation of character data during insert and update processing.

Default value: NO

**Details**

For a full description of this option, refer to “DBFORCE=” on page 43.

Note: This option does not override the FORCE statement in the APPEND procedure.

**DBINDEX=**

Indicates whether SAS calls the DBMS to find indexes on the specified table.

Default value: Defaults to the value of the LIBNAME option.

**Details**

For a full description of this option, refer to “SAS/ACCESS Data Set Options” on page 39.

**DBLINK=**

Specifies an object in the local database that allows access to objects in a remote database.
Default value: Defaults to the value of the LIBNAME option.

**Syntax**

```plaintext
DBLINK=database-link
```

**database-link**

specifies a link to a remote object, such as a table or view in another database.

**Details**

If you specify a link, SAS uses the link to access remote objects. If you omit \texttt{DBLINK=}, SAS accesses objects in the local database.

A link is a database object that identifies an object that is stored in a remote database. A link contains stored path information and may also contain user name and password information for connecting to the remote database.

**Example: Referencing a Remote ORACLE Table By Using DBLINK=**

In this example, SAS sends \texttt{myoradb.employees} to ORACLE as \texttt{emp@sales.hq.acme.com}.

```sas
proc print data=myoradb.employees(dblink= 'sales.hq.acme.com');
run;
```

**DBNULL=**

Indicates whether NULL is a valid value for the specified column(s).

Default value: YES

**Details**

For a full description of this option, refer to “DBNULL=” on page 47.

**DBPROMPT=**

Specifies whether SAS displays a prompting window that enables you to enter SAS/ACCESS engine connection options instead of specifying them in the LIBNAME statement.

Default value: Defaults to the value of the LIBNAME option.

**Details**

If you omit \texttt{DBPROMPT=}, the default value is \texttt{DBPROMPT=NO}. The SAS/ACCESS Interface to ORACLE allows you to enter 30 characters each for the \texttt{USERNAME} and
PASSWORD and up to 70 characters for the PATH, depending on your platform and terminal type.

Note: This option is appropriate only for view descriptors created in Version 6. For a full description of this option, refer to “DBPROMPT=” on page 48.

**DBSASTYPE=**

Specifies data type(s) to override the default SAS data type(s) during input processing of data from ORACLE.

**Default value:** Varies by data type.

**Syntax**

```
DBSASTYPE=(<column-name-1='SAS-data-type'>
   <...<column-name-n='SAS-data-type'>>)
```

- **column-name** specifies a DBMS column name.
- **SAS-data-type** specifies a SAS data type.

**Details**

This option is valid only when you read ORACLE data into SAS. By default, the SAS/ACCESS Interface to ORACLE converts each ORACLE data type to a predetermined SAS data type when processing data from ORACLE. When you need a different data type, you can use DBSASTYPE= to override the default data type chosen by the SAS/ACCESS engine.

In the following example, DBSASTYPE= specifies a data type to use for the column MYCOLUMN when printing the DBMS data in SAS. If the data in this DBMS column is stored in a format that SAS does not support, such as NUMBER(20), this enables SAS to print the values.

```
proc print data=mylib.mytable
   (DBSASTYPE=(mycolumn=$20.));
run;
```

See “LIBNAME Statement Data Conversions” on page 399 for more details on the default data types for ORACLE.

**DBTYPE=**

Specifies data type(s) to override the default ORACLE data type(s) when SAS outputs data to ORACLE.
Default value: VARCHAR2(size) for characters, where size is derived from the SAS variable length; NUMBER(p, s) for numbers, where p and s are derived from the SAS variable format; and NUMBER for numbers where p and s cannot be derived.

Syntax

DBTYPE=(<column-name-1=><DBMS-type>><column-name-n=><DBMS-type><>)

Details

For a full description of this option, refer to “DBTYPE=” on page 49.

---

**INSERTBUFF=**

Specifies the number of rows in a single ORACLE insert.

Default value:  Defaults to the value of the LIBNAME option.

Syntax

INSERTBUFF=positive-integer

positive-integer

specifies the number of rows to insert.

Details

SAS allows the maximum that is allowed by ORACLE.

Note:  When you assign a value that is greater than INSERTBUFF=1, the SAS application notes that indicate the success or failure of the insert operation may be incorrect because these notes only represent information for a single insert, even when multiple inserts are performed.  

Note:  If specified, the value of the DBCOMMIT= option overrides the value of INSERTBUFF=.

---

**ORHINTS=**

Specifies ORACLE hints to pass to ORACLE from a SAS statement or SQL procedure.

Default value:  None
Syntax

**ORHINTS** =ORAICL-hint’

**ORAICL-hint**

specifies an ORACLE hint.

Details

If you specify an ORACLE hint, SAS passes the hint to ORACLE. If you omit ORHINTS=, SAS does not send any hints to ORACLE.

ORHINTS= can also be specified with the HINTS= alias.

Example: Passing an ORACLE Hint in a SAS PROC step

When you run a SAS procedure on DBMS data, SAS converts the procedure to one or more SQL queries. The ORHINTS= data set option enables you to specify an ORACLE hint for SAS to pass as part of the SQL query.

```
libname mydblib oracle user=scott password=tiger
   path='myorapath';

proc print data=mydblib.payroll(orhints=
   ‘/*+ ALL_ROWS */’);
run;
```

In this example, SAS sends the ORACLE hint ‘/*+ ALL_ROWS */’ to ORACLE as part of the following statement:

```
SELECT /*+ ALL_ROWS */ * FROM PAYROLL
```

**READ_ISOLATION_LEVEL=**

Specifies which isolation level to use when reading ORACLE tables and views.

Default value:  Defaults to the value of the LIBNAME option.

Syntax

READ_ISOLATION_LEVEL=READCOMMITTED | SERIALIZABLE

**READCOMMITTED**

instructs SAS to use the read committed isolation level when reading ORACLE tables.

**SERIALIZABLE**

instructs SAS to use the serializable isolation level when reading ORACLE tables.
Details

The SPOOL= option overrides the READ_ISOLATION_LEVEL= option. The ORACLE_73= option must be set to ORACLE_73=YES to use the SERIALIZABLE isolation level.

If SPOOL=DBMS, SAS automatically sets READ_ISOLATION_LEVEL=SERIALIZABLE for ORACLE versions 7.3 or later and READ_ONLY for prior versions. In all other cases, the default value is READCOMMITTED.

When READ_ISOLATION_LEVEL is set to SERIALIZABLE, the CONNECTION option must be set to UNIQUE. If not, an error occurs.

Note: This option should be rarely needed, since the SAS/ACCESS Interface to ORACLE chooses the appropriate isolation level based on other locking options.

READ_LOCK_TYPE=

Specifies how a table is locked during read operations.

Default value: Defaults to the value of the LIBNAME option.

Details

If you specify READ_LOCK_TYPE=NOLOCK, table locking is not used when the data set is read. If you specify READ_LOCK_TYPE=ROW, the ORACLE “ROW SHARE” table lock is used when the data set is read. If you specify READ_LOCK_TYPE=TABLE, the ORACLE “SHARE” table lock is used when the data set is read.

When READ_LOCK_TYPE is set to either TABLE or ROW, the CONNECTION option must be set to UNIQUE. If not, an error occurs.

For a full description of this option, refer to “READ_LOCK_TYPE=” on page 52.

READBUFF=

Specifies the number of rows in a single ORACLE fetch.

Default value: Defaults to the value of the LIBNAME option.

Syntax

READBUFF=positive-integer

Details

SAS allows the maximum that is allowed by ORACLE.
READBUFF= can also be specified with the BUFFSIZE= alias.

SASDATEFMT=

Specifies the ORACLE date format to use to convert SAS date values.
Default value: DATETIME20.0

Details
For a full description of this option, refer to “SAS/ACCESS Data Set Options” on page 39.

SCHEMA=

Specifies a schema name to be used when referring to any database object.
Default value: Defaults to the value of the LIBNAME option.

Syntax
SCHEMA=schema-name

schema-name
  specifies a schema name.

Details
If PRESERVE_TAB_NAMES=NO, SAS converts the SCHEMA= value to uppercase because all values in the ORACLE data dictionary are converted to uppercase unless quoted.

Example: Accessing an ORACLE table using SCHEMA=
In this example, SAS sends any reference to employees as scott.employees.
  proc print data=employees schema=scott;
  run;

UPDATE_ISOLATION_LEVEL=

Specifies which isolation level to use when reading ORACLE tables for update.
Default value: Defaults to the value of the LIBNAME option.
UPDATE_LOCK_TYPE=

Syntax

**UPDATE_ISOLATION_LEVEL**=READCOMMITTED | SERIALIZABLE

**READCOMMITTED**

instructs SAS to use the read committed isolation level when reading ORACLE tables for update.

**SERIALIZABLE**

instructs SAS to use the serializable isolation level when reading ORACLE tables for update.

Details

The ORACLE_73= option must be set to ORACLE_73=YES to use the SERIALIZABLE method.

If UPDATE_LOCK_TYPE=NOLOCK, SAS automatically sets UPDATE_ISOLATION_LEVEL=SERIALIZABLE for ORACLE versions 7.3 or later and READ_ONLY for prior versions. In all other cases, the default value is READCOMMITTED.

Note: This option should be rarely needed, since the SAS/ACCESS Interface to ORACLE chooses the appropriate isolation level based on other locking options.

UPDATE_LOCK_TYPE=

Specifies how a table is locked during update operations.

Default value: Defaults to the value of the LIBNAME option.

Details

If you specify UPDATE_LOCK_TYPE=NOLOCK, table locking is not used during the reading of tables and views for update. If you specify UPDATE_LOCK_TYPE=ROW, the ORACLE “ROW SHARE” table lock is used when the data set is read for update. If you specify UPDATE_LOCK_TYPE=TABLE, the ORACLE “EXCLUSIVE” table lock is used when the table is read for update.

If UPDATE_LOCK_TYPE=NOLOCK and ORACLE_73_OR_ABOVE=YES, updates are performed using serializable transactions. If UPDATE_LOCK_TYPE=NOLOCK and ORACLE_73_OR_ABOVE=NO, updates are performed using an extra WHERE clause to ensure that the row was not updated since it was first read. Updates might fail when UPDATE_LOCK_TYPE=NOLOCK because other users might modify a row after the row was read for update.

If the ORACLE_73_OR_ABOVE= option is incorrectly set to YES (meaning that the Oracle server version is below 7.3), the Oracle engine detects this, and the update is performed as if ORACLE_73_OR_ABOVE= were correctly set.

For a full description of this option, refer to “UPDATE_LOCK_TYPE=” on page 53.
**UPDATEBUFF=**

Specifies the number of rows in a single ORACLE update/delete.

Default value: Defaults to the value of the LIBNAME option.

---

**Syntax**

```
UPDATEBUFF=positive-integer
```

`positive-integer` specifies a number of rows.

**Details**

SAS allows the maximum that is allowed by ORACLE.

---

**ACCESS Procedure: ORACLE Specifics**

The following section describes the DBMS-specific statements that you use in the SAS/ACCESS Interface to ORACLE.

---

**ACCESS Procedure Statements for ORACLE**

To create an access descriptor, you use the `DBMS=ORACLE` option and the database description statements `PATH=`, `ORAPW=`, `USER=`, and `TABLE=` in the `PROC ACCESS` step. The database description statements supply DBMS-specific information to the SAS System. These statements must immediately follow the `CREATE` statement that specifies the access descriptor to be created.

Database description statements are required only when you create access descriptors. Because ORACLE information is stored in an access descriptor, you do not need to repeat this information when you create view descriptors.

The SAS/ACCESS Interface to ORACLE uses the following procedure statements in line or batch mode:

```
PROC ACCESS <DBMS=ORACLE | view-descriptor-options>
   CREATE libref.member-name.ACCESS | VIEW <password-option>;
   UPDATE libref.member-name.ACCESS | VIEW <password-option>;
   USER= <'ORACLE-user-name'>;
   ORAPW= | ORACLEPW=<'ORACLE-password'>;
   TABLE= <'ORACLE-table-name'>;
   PATH= 'ORACLE-path-designation';
   ASSIGN <= YES | NO | Y | N;
   DROP <'column-identifier-1'> <...<'column-identifier-n'>>
```
**ACCESS Procedure Examples**

The following example creates an access descriptor and a view descriptor based on ORACLE data.

```plaintext
FORMAT <'column-identifier-1'='SAS-format-name-1
   ...<>'column-identifier-n'='SAS-format-name-n'>;
QUIT;
RENAME <'column-identifier-1'='SAS-variable-name-1
   ...<>'column-identifier-n'='SAS-variable-name-n'>;
RESET ALL | <'column-identifier-1' ... <'column-identifier-n'>;
SELECT ALL | <'column-identifier-1' ... <'column-identifier-n'>;
SUBSET selection-criteria;
UNIQUE => YES | NO | Y | N;
LIST <ALL | VIEW | <'column-identifier'>;
RUN ;

USER= <'ORACLE-user-name'>;

specifies an optional ORACLE user name. If the user name contains blanks or national characters, enclose the name in quotation marks. See “ORACLE Naming Conventions” on page 396 for more information.

If you omit an ORACLE user name and password, the default ORACLE user ID OPSSysid is used, if it is enabled. USER= must be used with ORAPW=.

ORAPW= | ORACLEPW= <'ORACLE-password'>;

specifies an optional ORACLE password that is associated with the ORACLE user name. If omitted, the password for the default ORACLE user ID OPSSysid is used, if it is enabled. ORAPW= must be used with USER=.

TABLE=<'ORACLE-table'>;

specifies the name of the ORACLE table or ORACLE view on which the access descriptor is based. This statement is required.

The ORACLE-table-name argument can be up to 30 characters long and must be a valid ORACLE table name. If the table name contains blanks or national characters, enclose the name in quotation marks.

PATH=<'ORACLE-database-specification'>;

specifies the ORACLE driver, node, and database. Aliases are required if you are using SQL*Net Version 2.0 or later. In some operating environments, you can enter the information that is required by the PATH= statement before invoking the SAS System.

SAS/ACCESS uses the same ORACLE path designation that you use to connect to ORACLE directly. See your database administrator to determine the databases that have been set up in your operating environment, and to determine the default values if you do not specify a database. On UNIX systems, the TWO_TASK environment variable is used, if set. If neither PATH= nor TWO_TASK have been set, the default value is the local driver.

The values that you specify for the USER=, ORAPW=, TABLE=, and PATH= statements are permanently associated with the access descriptor that you create, and with all the view descriptors that are created from that access descriptor.

---

**ACCESS Procedure Examples**

The following example creates an access descriptor and a view descriptor based on ORACLE data.
The following section describes the DBMS-specific statements that you use in the SAS/ACCESS interface to ORACLE.
**DBLOAD Procedure Statements for ORACLE**

To create and load an ORACLE table, the SAS/ACCESS Interface to ORACLE uses the following statements in interactive-line or batch mode.

```plaintext
PROC DBLOAD DBMS=ORACLE <DATA= <libref.>SAS-data-set> <APPEND>;  
   TABLE= '<>ORACLE-table-name<>';  
   USER= '<>ORACLE-user-name<>';  
   ORAPW= | PASS= | PW= | PASSWORD= '<>ORACLE-password<>';  
   PATH= '<>ORACLE-database-specification<>';  
   TABLESPACE= '<>ORACLE-tablespace-name<>';  
   ACCDESC= <libref.>access-descriptor;  
   COMMIT= commit-frequency;  
   DELETE variable-identifier-1 <...variable-identifier-n>;  
   ERRLIMIT= error-limit;  
   LABEL;  
   LIMIT=load-limit;  
   NULLS variable-identifier-1 = Y | N <...variable-identifier-n = Y | N>;  
   QUIT;  
   RENAME variable-identifier-1 = '<>column-name1<>'  
      <... variable-identifier-n = '<>column-name-n<>';  
   RESET ALL | variable-identifier-1 <...variable-identifier-n>;  
   SQL ORACLE-SQL-statement;  
   TYPE variable-identifier-1 = 'column-type-1'  
      <...variable-identifier-n = 'column-type-n'>;  
   WHERE SAS-where-expression;  
   LIST <ALL | COLUMN | variable-identifier>;  
   LOAD;  
   RUN;
```

**PROC DBLOAD Statements**

- **TABLE=<>ORACLE-table-name<>;**
  - identifies the ORACLE table that you want to create. The TABLE= statement is required.
  - The ORACLE-table-name argument can be up to 30 characters long and must be a valid ORACLE table name. If the table name contains blanks or national characters, enclose the name in quotation marks.

- **USER=<>ORACLE-user-name<>;**
  - specifies an optional ORACLE userid. If the user name contains blanks or national characters, enclose the name in quotation marks. If you omit an ORACLE user name and password, the default ORACLE user ID OPS$sysid is used, if it is enabled. USER= must be used with ORAPW=

- **ORAPW= | PW= | PASS= | PASSWORD= <>ORACLE-password<>;**
  - specifies an optional ORACLE password that is associated with the ORACLE user ID that is specified in the USER= statement. If omitted, the password for the
default ORACLE user ID OPS$sysid is used, if it is enabled. ORAPW= must be used with USER=.

**TABLESPACE=</ORACLE-tablespace-name>**

specifies the name of the ORACLE tablespace where you want to store the new table.

The ORACLE-tablespace-name argument can be up to 18 characters long and must be a valid ORACLE tablespace name. If the name contains blanks or national characters, enclose the entire name in quotation marks.

If omitted, the table is created in the user's default tablespace that is defined by the ORACLE database administrator at your site.

**PATH=</ORACLE-database-specification>**

specifies the ORACLE driver, node, and database. Aliases are required if you are using SQL*Net Version 2.0 or later. In some operating environments, you can enter the information that is required by the PATH= statement before invoking the SAS System.

SAS/ACCESS uses the same ORACLE path designation that you use to connect to ORACLE directly. See your database administrator to determine the path designations that have been set up in your operating environment, and to determine the default value if you do not specify a path designation. On UNIX systems, the TWO_TASK environment variable is used, if set. If neither PATH= nor TWO_TASK have been set, the default value is the local driver.

---

**DBLOAD Procedure Examples**

The following example creates a new ORACLE table, EXCHANGE, from the DLIB.RATEOFEX data file. An access descriptor, ADLIB.EXCHANGE, based on the new table, is also created. The PATH= statement uses an alias to connect to a remote ORACLE 7 Server database.

The SQL statement in the second DBLOAD procedure sends an SQL GRANT statement to ORACLE. You must be granted ORACLE privileges to create new ORACLE tables or to grant privileges to other users. The SQL statement is in a separate procedure because you cannot create a DBMS table and reference it within the same DBLOAD step. The new table is not created until the RUN statement is processed at the end of the first DBLOAD step.

Note: The DLIB.RATEOFEX data set is included in the sample data shipped with your software.

```
libname adlib 'SAS-data-library';
libname dlib 'SAS-data-library';

proc dbload dbms=oracle data=dlib.rateofex;
  user=scott; orapw=tiger;
  path='myorapath';
  table=exchange;
  accdesc=adlib.exchange;
  rename fgnindol=fgnindolar 4=dolrsinfgn;
  nulls updated=n fgnindol=n 4=n country=n;
  load;
```
The next example uses the APPEND option to append rows from the INVDATA data set to an existing ORACLE table named INVOICE.

proc dbload dbms=oracle data=invdata append;
  user=scott; orapw=tiger;
  path='myorapath';
  table=invoice;
  load;
run;

Note: The next example uses a previously created data set, INVDATA.

SQL Procedure Pass-Through Facility: ORACLE Specifics

The following section describes the DBMS-specific arguments that you use in the CONNECT statement to establish a connection with an ORACLE database.

Arguments to Connect to ORACLE

The CONNECT statement is optional when connecting to ORACLE. If you omit the CONNECT statement, an implicit connection is made with your OPS$sysid, if it is enabled. When you omit a CONNECT statement, an implicit connection is performed when the first EXECUTE statement or CONNECTION TO component is passed to ORACLE. In this case you must use the default DBMS name ORACLE. The interface to ORACLE can connect to multiple databases (both local and remote) and to multiple user IDs. If you use multiple simultaneous connections, you must use an alias argument to identify each connection. If you do not specify an alias, the default alias ORACLE is used.

CONNECT TO ORACLE <AS alias>
  (USER=ORACLE-user-name
   PASSWORD=ORACLE-password
   PATH="ORACLE-path-designation"
   BUFFSIZE=number-of-rows
   PRESERVE_COMMENTS);
USER=<>ORACLE-user-name<> specifies an optional ORACLE user name. If you omit an ORACLE password and user name, the default ORACLE user ID OPS$sysid is used if it is enabled. If you specify USER=, you must also specify ORAPW=.

ORAPW= | PASSWORD= | PASS= | PW= <>ORACLE-password<> specifies an optional ORACLE password that is associated with the ORACLE user name. If you omit an ORACLE password, the default ORACLE user ID OPS$sysid is used, if it is enabled. If you specify ORAPW=, you must also specify USER=.

BUFFSIZE=number-of-rows specifies the number of rows to retrieve from an ORACLE table or view with each fetch. Using this argument can improve the performance of any query to ORACLE.

    By setting the value of the BUFFSIZE= argument in your SAS programs, you can find the optimal number of rows for a given query on a given table. The default buffer size is 25 rows per fetch. The limit is 32,767 rows per fetch, although a practical limit for most applications is less, depending on the available memory.

PRESERVE_COMMENTS enables you to pass additional information (called “hints”) to ORACLE for processing. These hints might direct the ORACLE query optimizer to choose the best processing method based on your hint.

    You specify PRESERVE_COMMENTS as an argument in the CONNECT statement. Then you specify the hints in the CONNECTION TO component’s ORACLE SQL query. The hints are entered as comments in the SQL query and are passed to and processed by ORACLE.

PATH=<>ORACLE-database-specification<> specifies the ORACLE driver, node, and database. Aliases are required if you are using SQL*Net Version 2.0 or later. In some operating environments, you can enter the information that is required by the PATH= statement before invoking the SAS System.

    SAS/ACCESS uses the same ORACLE path designation that you use to connect to ORACLE directly. See your database administrator to determine the path designations that have been set up in your operating environment, and to determine the default value if you do not specify a path designation. On UNIX systems, the TWO_TASK environment variable is used, if set. If neither PATH= nor TWO_TASK have been set, the default value is the local driver.

---

**Pass-Through Examples**

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

```
proc sql;
    connect to oracle as dbcon
        (user=scott pass=tiger buffsize=100
             path='myorapath');
quit;
```
The following example connects to ORACLE and sends it two EXECUTE statements to process.

```sql
proc sql;
    connect to oracle (user=scott
                  password=tiger);

    execute (create view whotookorders as
              select ordernum, takenby,
                   firstname, lastname, phone
              from orders, employees
              where orders.takenby=employees.empid)
           by oracle;
    execute (grant select on whotookorders
               to testuser) by oracle;

    disconnect from oracle;
quit;
```

The following example performs a query, shown in underlined text, on the ORACLE table CUSTOMERS:

```sql
proc sql;
    connect to oracle (user=scott
                  password=tiger);
    select *
     from connection to oracle
        (select * from customers
         where customer like '1%');
    disconnect from oracle;
quit;
```

In this example, the PRESERVE_COMMENTS argument is specified after the USER= and ORAPW= arguments. The ORACLE SQL query is enclosed in required parentheses. The SQL INDEX command identifies the index for the ORACLE query optimizer to use in processing the query. Note that multiple hints are separated with blanks.

```sql
proc sql;
    connect to oracle as mycon(user=scott
                  password=tiger preserve_comments);
    select *
     from connection to mycon
        (select /* +indx(empid) all_rows */ count(*) from employees);
quit;
```

---

**ORACLE Naming Conventions**

ORACLE objects that may be named include tables, views, columns, and indexes. For the ORACLE7 Server, objects also include database triggers, procedures, and stored functions. Use the following ORACLE naming conventions:

- A name must start with a letter. However, if the name appears within double quotation marks, it may start with any character.
A name must be from 1 to 30 characters long, except for database names, which are limited to 8 characters, and link names, which are limited to 128 characters.

A name may contain the letters A through Z, the digits 0 through 9, the underscore (_), $, and #. If the name appears within double quotation marks, it may contain any characters, except double quotation marks.

A name is not case sensitive. For example, CUSTOMER is the same as customer. If, however, the name of the object appears within double quotation marks when it is used, then it is case sensitive.

A name cannot be an ORACLE reserved word.

A name cannot be the same name as another ORACLE object.

ORACLE7 Server Data Types

Every column in a table has a name and a data type. The data type tells ORACLE how much physical storage to set aside for the column and the form in which the data is stored. ORACLE data types fall into three categories: types for character data, types for numeric data, and types for abstract values such as dates. Each of these types is described in the following sections.

Note: The SAS/ACCESS Interface to ORACLE does not support the following ORACLE data types: MLSLABEL and ROWID. SAS/ACCESS provides an error message when it attempts to read a table that has at least one column that uses an unsupported data type.

Character Data

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. Note that the ORACLE7 Server CHAR data type is not equivalent to the ORACLE Version 6 CHAR data type. The ORACLE7 Server CHAR data type is new with ORACLE7 Server and uses blank-padded comparison semantics.

LONG

contains varying-length character string data that is similar to type VARCHAR2. Type LONG is character data of variable length with a maximum length of 2 gigabytes. You can define only one LONG column per table. Available memory considerations might also limit the size of a LONG data type.

VARCHAR2(n)

contains character string data with a length of n, where n must be at least 1 and cannot exceed 2000 characters. The VARCHAR2 data type is equivalent to the ORACLE Version 6 CHAR data type except for the difference in maximum lengths. The VARCHAR2 data type uses nonpadded comparison semantics.
**Numeric Data**

**NUMBER(p,s)**

specifies a fixed-point number with an implicit decimal point, where \( p \) is the total number of digits (precision) and can range from 1 to 38, and \( s \) is the number of digits to the right of the decimal point (scale) and can range from -84 to 127.

**NUMBER(p)**

specifies an integer of precision \( p \) that can range from 1 to 38 and a scale of 0.

**NUMBER**

specifies a floating-point number with a precision of 38. A floating-point value can either specify a decimal point anywhere from the first to the last digit or omit the decimal point. A scale value does not apply to floating-point numbers since there is no restriction on the number of digits that can appear after the decimal point.

**Abstract Data**

**DATE**

contains date values. Valid dates are from January 1, 4712 BC to December 31, 4712 AD. The default format is DD-MON-YY, for example '05-OCT-98'.

**LONG RAW**

contains raw binary data of variable length up to 2 gigabytes. Values entered into columns of this type must be inserted as character strings in hexadecimal notation.

**RAW(n)**

contains raw binary data where \( n \) must be at least 1 and cannot exceed 255 bytes. Values entered into columns of this type must be inserted as character strings in hexadecimal notation. You must specify \( n \) for this data type.

**Note** For compatibility with other DBMSs, ORACLE supports the syntax for a wide variety of numeric data types, including DECIMAL, INTEGER, REAL, DOUBLE-PRECISION, and SMALLINT. All forms of numeric data types are actually stored in the same internal ORACLE NUMBER format. The additional numeric data types are variations of precision and scale. A null scale implies a floating-point number, and a non-null scale implies a fixed-point number.

See “LIBNAME Statement Data Conversions” on page 399 and “ACCESS Procedure Data Conversions” on page 400 for a description of how PROC ACCESS and the LIBNAME statement treat each of these types during input operations.

**NULL and Default Values**

ORACLE has a special value called NULL. NULL means an absence of information and is analogous to the SAS System's missing value. By default, columns accept NULL values. However, you can define columns so that they cannot contain NULL data. For example, the CREATE TABLE statement for the CUSTOMERS table in Appendix 1 defines the first column, CUSTOMER, as CHAR(8) and NOT NULL. NOT NULL tells ORACLE not to add a row to the table unless the row has a value for CUSTOMER.
LIBNAME Statement Data Conversions

Table 20.1 on page 399 shows the default SAS System variable formats that the LIBNAME statement assigns to ORACLE data types during input operations. You can override these input and output data types by using the DBTYPE= data set option during output processing.

Note: ORACLE data types that are omitted from this table are not supported by SAS/ACCESS.

Table 20.1 LIBNAME Statement: Default SAS Formats for ORACLE Data Types

<table>
<thead>
<tr>
<th>ORACLE Data Type</th>
<th>Default SAS Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR(n)</td>
<td>$n.</td>
</tr>
<tr>
<td>VARCHAR2(n)</td>
<td>$n.</td>
</tr>
<tr>
<td>NUMBER</td>
<td>none (BEST. on MVS)</td>
</tr>
<tr>
<td>NUMBER(p)</td>
<td>w.(BEST. on MVS)</td>
</tr>
<tr>
<td>NUMBER(p, s)</td>
<td>w.d</td>
</tr>
<tr>
<td>DATE</td>
<td>DATETIME.20.</td>
</tr>
<tr>
<td>LONG</td>
<td>$32,767.</td>
</tr>
<tr>
<td>RAW(n)</td>
<td>$HEXw.</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>$HEX32,767.</td>
</tr>
</tbody>
</table>

If ORACLE data falls outside valid SAS data ranges, the values are usually counted as missing.

Note: SAS automatically converts ORACLE NUMBER types to SAS number formats by using an algorithm that determines the correct scale and precision. When the scale and precision cannot be determined, the SAS/ACCESS Interface to ORACLE allows the procedure or application to determine the format.

Note: Reading an ORACLE table that contains a column of data type LONG produces a great deal of output. Use caution when working with columns of this type.

Table 20.2 on page 399 shows the default ORACLE data types that the LIBNAME statement assigns to SAS variable formats during output operations.

Table 20.2 LIBNAME Statement: Default ORACLE Data Types for SAS Formats

<table>
<thead>
<tr>
<th>SAS Variable Format</th>
<th>ORACLE Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w.</td>
<td>VARCHAR2(n)</td>
</tr>
<tr>
<td>w. with SAS format name of NULL</td>
<td>NUMBER(p)</td>
</tr>
<tr>
<td>w.d with SAS format name of NULL</td>
<td>NUMBER(p,s)</td>
</tr>
<tr>
<td>all other numerics *</td>
<td>NUMBER (NUMBER(38,10) on MVS)</td>
</tr>
<tr>
<td>datetimew.d</td>
<td>DATE</td>
</tr>
</tbody>
</table>
ACCESS Procedure Data Conversions

Table 20.3 on page 400 shows the default SAS System variable formats that the ACCESS procedure assigns to ORACLE data types.

Note: ORACLE data types that are omitted from this table are not supported by the SAS/ACCESS Interface.

Table 20.3  PROC ACCESS: Default SAS Formats for ORACLE Data Types

<table>
<thead>
<tr>
<th>ORACLE Data Type</th>
<th>Default SAS Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR(n)</td>
<td>$n. (n &lt;= 200) $200. (n &gt; 200)</td>
</tr>
<tr>
<td>VARCHAR2(n)</td>
<td>$n. (n &lt;= 200) $200. (n &gt; 200)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>BEST22.</td>
</tr>
<tr>
<td>NUMBER</td>
<td>BEST22.</td>
</tr>
<tr>
<td>NUMBER(p)</td>
<td>w.</td>
</tr>
<tr>
<td>NUMBER(p, s)</td>
<td>w.d</td>
</tr>
<tr>
<td>DATE</td>
<td>DATETIME16.</td>
</tr>
<tr>
<td>LONG</td>
<td>$200.</td>
</tr>
<tr>
<td>RAW(n)</td>
<td>$n. (n &lt; 200) $200. (n &gt; 200)</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>$200.</td>
</tr>
</tbody>
</table>

See “ACCESS Procedure Data Conversions for the NUMBER Data Type” for more information about NUMBER data type conversions. If ORACLE data fall outside valid SAS data ranges, the values are usually counted as missing.

ACCESS Procedure Data Conversions for the NUMBER Data Type

The general form of an ORACLE number is NUMBER(p,s) where p is the precision and s is the scale of the number. ORACLE defines precision as the total number of digits, with a valid range of -84 to 127. However, a negative scale means that the number is rounded to the specified number of places to the left of the decimal. For example, if the number 1,234.56 is specified as data type NUMBER(8,-2), it is rounded to the nearest hundred and stored as 1,200.

Table 20.4 on page 401 shows the correlation between the ORACLE NUMBER data types and the default SAS formats that are created from that data type.
**Table 20.4** Default SAS Formats for ORACLE NUMBER Data Types

<table>
<thead>
<tr>
<th>ORACLE NUMBER Data Type</th>
<th>Rules</th>
<th>Default SAS Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER(p)</td>
<td>0 &lt; $p$ &lt;= 32</td>
<td>($p + 1$).0</td>
</tr>
<tr>
<td>NUMBER(p,s)</td>
<td>$p &gt; 0$, $s &lt; 0$, $</td>
<td>s</td>
</tr>
<tr>
<td>NUMBER(p,s)</td>
<td>$p &gt; 0$, $s &lt; 0$, $</td>
<td>s</td>
</tr>
<tr>
<td>NUMBER(p,s)</td>
<td>$p &gt; 0$, $s &gt; 0$, $s &lt; p$</td>
<td>($p + 2$).s</td>
</tr>
<tr>
<td>NUMBER(p,s)</td>
<td>$p &gt; 0$, $s &gt; 0$, $s &gt;= p$</td>
<td>($s + 3$).s</td>
</tr>
<tr>
<td>NUMBER(p)</td>
<td>$p &gt; 32$</td>
<td>BEST22. SAS selects format</td>
</tr>
<tr>
<td>NUMBER</td>
<td>$p,s$ unspecified</td>
<td>BEST22. SAS selects format</td>
</tr>
</tbody>
</table>

**DBLOAD Procedure Data Conversions**

Table 20.5 on page 401 shows the default ORACLE data types the DBLOAD procedure assigns to SAS variable formats.

**Table 20.5** PROC DBLOAD: Default ORACLE Data Types for SAS Formats

<table>
<thead>
<tr>
<th>SAS Variable Format</th>
<th>ORACLE Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w.$</td>
<td>CHAR(n)</td>
</tr>
<tr>
<td>w.</td>
<td>NUMBER(p)</td>
</tr>
<tr>
<td>w.d</td>
<td>NUMBER(p,s)</td>
</tr>
<tr>
<td>all other numerics *</td>
<td>NUMBER</td>
</tr>
<tr>
<td>datetime w.d</td>
<td>DATE</td>
</tr>
<tr>
<td>date w.</td>
<td>DATE</td>
</tr>
<tr>
<td>time. **</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

* Includes all SAS numeric formats, such as BINARY8 and E10.0.
** Includes all SAS time formats, such as TODw.d and HHMMw.d.