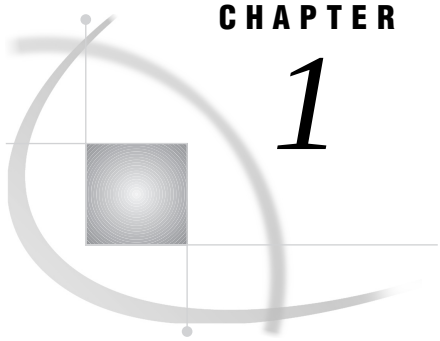

CHAPTER

1

Oracle Rdb Chapter, First Edition

Introduction

This chapter introduces SAS System users to Oracle Rdb, which is a relational database management system (DBMS). It accompanies and should be used with *SAS/ACCESS Software for Relational Databases: Reference* (order #57204).*

This chapter focuses on terms and concepts that help you use the SAS/ACCESS Interface to Oracle Rdb. It describes the Oracle Rdb-specific options and statements that you use in the ACCESS and DBLOAD procedures and in the SQL procedure's CONNECT statement.

For general information about database management systems, including information for the database administrator about how the SAS/ACCESS interfaces work, refer to Appendix 2, "DBMS Overview and Information for the Database Administrator". For more information on an Oracle Rdb concept or term, see your Oracle Rdb documentation.

Version 7 and Version 8 Information

The SAS/ACCESS LIBNAME and data set options in Version 7 and Version 8 of SAS/ACCESS software do not apply to Oracle Rdb at this time. However, you can use the ACCESS and DBLOAD procedures to access or load DBMS data. You can still create SAS/ACCESS descriptor files and PROC SQL Pass-Through queries and views. There are some changes and enhancements to the ways in which these features work:

- The SAS Explorer window replaces the ACCESS window and its functionality.
- The ACCESS procedure has been enhanced by the UPDATE statement, which enables you to edit and update access descriptors and view descriptors.

Data Set Options: Oracle Rdb Specifics

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

RDBLOCK= on page 2
RDBCONST= on page 3

Multiple-User Update Access to Oracle Rdb Tables (RDBLOCK=)

In Releases 6.08 and later of the SAS System, you can use the RDBLOCK= data set option, which allows multiple users to update data in a table concurrently. This SAS option can be used only with the FSEDIT procedure.

If you omit the RDBLOCK= option, the default value, RDBLOCK=NOCONCUR, is used. A table-level lock is put on the table, which means that only one user can update the table at a time.

Specify RDBLOCK=CONCUR for multiple-user, concurrent update access when you use a view descriptor that updates Oracle Rdb data. For example:

```
proc fsedit data=vlib.usacust(rdblock=concur);
run;
```

Note: This example references a previously created view descriptor named VLIB.CUSTOMER. △

In this example, an update or deletion of a row locks that row (or record) until the commit is issued, but the rest of the Oracle Rdb table remains accessible to other users for additional updates or deletions. This is known as *record-level locking*.

If you specify a view descriptor in PROC FSEDIT and you are not allowed to update a row, Oracle Rdb writes one of the following error messages to the SAS log:

- **ERROR: Update failed. %RDB-E-LOCK_CONFLICT, request failed due to locked resource.**

  REASON: Another user has locked this row in order to update it.

  ACTION: You can wait until the user releases the lock, or you can type CANCEL on the FSEDIT command line to cancel the update and advance to the next row.

- **ERROR: Update failed. %SQL-W-NOTFOUND, No rows were found for this statement.**

  REASON: Another user changed the values of the row after you read the values that are displayed on your screen. The WHERE expression that the interface view

engine appended to the Oracle Rdb SQL UPDATE statement (to ensure the data integrity of your update) failed to find the appropriate rows to update.

ACTION: Type CANCEL on the FSEDIT command line to cancel the update. Refresh the row by scrolling forward to the next row, then scroll backward to the observation that you attempted to update.

Note: When RDBLOCK=NOCONCUR is in effect, updates, deletions, and insertions are committed after you type the FSEDIT SAVE command and after you end the FSEDIT procedure. When RDBLOCK=CONCUR is in effect, updates, deletions, and insertions are committed after you type FSEDIT SAVE, after you end the FSEDIT procedure, *and whenever you scroll backward.* △

You can control the amount of time that the system waits for update access before it provides an error message. To do so, set the logical lock timeout interval before invoking SAS. For example, the following DCL command sets the lock timeout interval to 5 seconds:

```
$define rdm$bind_lock_timeout_interval 5
```

Evaluating Constraints at Commit Time (RDBCONST=)

The RDBCONST= data set option enables you to override the Oracle Rdb default action of evaluating constraints at commit time. This option has no effect on constraints that must be evaluated at commit time.

The RDBCONST= option corresponds to the execution of the following Oracle Rdb SQL statements:

SAS Option Value	Oracle Rdb SQL Statement
RDBCONST=ON	SET DEFAULT CONSTRAINTS ON
RDBCONST=OFF	SET DEFAULT CONSTRAINTS OFF

The default setting is RDBCONST=OFF. This causes the constraint evaluation to be deferred until commit time, as in Releases 6.06 and 6.07 of the SAS/ACCESS Interface to Oracle Rdb.

When you use the default RDBCONST=OFF, a commit might not occur until several rows of data have been entered. Be aware that if RDBCONST=OFF and you enter a NULL value for a column that has been defined as NOT NULL, you are not notified of your error until commit time. All of the rows that are part of the current commit batch are rejected, and you must re-enter or modify all of the rows of data that you have entered since the previous commit.

Setting RDBCONST=ON causes all of the affected constraints to be evaluated after each row (or observation) is entered or updated. If your data violates a NULL constraint, you are informed immediately, and only the current record is rejected.

When RDBCONST=ON is specified, the SET DEFAULT CONSTRAINTS ON statement is executed immediately after you are connected to the database. It is re-executed after each commit because the statement remains in effect for only one transaction.

The following example shows how to specify the RDBCONST= option in a view descriptor that is based on Oracle Rdb data.

```
proc fsedit data=adlib.allinv(rdbconst=on);
run;
```

Note: This example references a previously created view descriptor named VLIB.INVOICE. △

ACCESS Procedure: Oracle Rdb Specifics

Chapter 9, "ACCESS Procedure Reference" describes the generic options and procedure statements that enable you to create access descriptors, view descriptors, and SAS data files from DBMS data. The following section describes the DBMS-specific statements that you use in the SAS/ACCESS Interface to Oracle Rdb.

ACCESS Procedure Statements for Oracle Rdb

To create an access descriptor, you can use the DATABASE= statement, which is a database-description statement for Oracle Rdb. Database-description statements supply database-specific information to the SAS System. If used, the DATABASE= statement must immediately follow the CREATE statement that specifies the access descriptor to be created. The DATABASE= statement is optional.

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

The syntax for other statements you use to create an access descriptor is provided below. For complete descriptions of these statements and their uses, see Chapter 9, "ACCESS Procedure Reference".

Note: Unlike other SAS/ACCESS products, the SAS/ACCESS Interface to Oracle Rdb does not use the following procedure statements: USER= and PASSWORD=. △

The SAS/ACCESS Interface to Oracle Rdb uses the following procedure statements in interactive line, noninteractive, or batch mode:

PROC ACCESS <*access-descriptor-options*|*view-descriptor-options*>;
ASSIGN <=> YES|NO;
CREATE <*libref.*>*member-name.*ACCESS|VIEW <*password-option*>;
UPDATE <*libref.*>*member-name.*ACCESS|VIEW <*password-option*>;

DATABASE=<'>*Oracle-Rdb-pathname*<'>;
DROP <'>*column-identifier-1*<'><...<'>*column-identifier-n*<'>>;
FORMAT<'>*column-identifier-1*<'><=> *SAS-format-name-1*
<...<'>*column-identifier-n*<'><=>*SAS-format-name-n*>;
LIST <ALL|VIEW|<'>*column-identifier*<'>>;
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*;
TABLE=<'>*Oracle-Rdb-table-name*<'>
UNIQUE<=>YES|NO;
RUN;

DATABASE=<'>*Oracle-Rdb-pathname*<'>;

The DATABASE= statement specifies the name and physical location of the Oracle Rdb database. The name can be the operating-system-specific pathname of the database or an OpenVMS logical name that points to the fully qualified Oracle Rdb database name. The .RDB file extension is optional.

If you are accessing a remote database, you can specify the OpenVMS node name as part of the OpenVMS pathname of the database:

DATABASE=<'>*<OpenVMS-net-node::> OpenVMS-pathname*<'>;

The DATABASE= statement is optional. If you omit the DATABASE= statement, the default action is to use the value of the OpenVMS logical name SQL$DATABASE. For more information about SQL$DATABASE and accessing remote databases, see your Oracle Rdb documentation.

Note: Double quotation marks cannot be used with this option. △

The following example uses a DATABASE= statement for a remote database called TEXTILE on the OpenVMS node ATLANTA.

```
proc access dbms=rdb;

   /*create access descriptor   */

   create adlib.customer.access;
   database='atlanta::disk1:[root]textile.rdb';
   table=customers;
   assign=yes;
   rename customer = customer_number
          firstorder = firstorder_date;
   list all;

   /*create usacust view        */

   create vlib.usacust.view;
   select customer state zipcode name
          firstorder;
   subset where customer like '1%';
run;
```

DBLOAD Procedure: Oracle Rdb Specifics

Chapter 10, "DBLOAD Procedure Reference" describes the generic options and procedure statements that enable you to create a DBMS table and to insert data in it. The following section describes the DBMS-specific statements that you use in the SAS/ACCESS Interface to Oracle Rdb.

DBLOAD Procedure Statements for Oracle Rdb

To create and load an Oracle Rdb table, you use the DBMS=RDB option and one optional database description statement in the PROC DBLOAD step, DATABASE=. The

database description statement supplies database-specific information to the SAS System.

The syntax for statements that you use to create and load an Oracle Rdb table is provided below. For complete descriptions of these statements and their uses, see Chapter 10, "DBLOAD Procedure Reference".

Note: Unlike other SAS/ACCESS products, the SAS/ACCESS interface to Oracle Rdb does not use the following procedure statements: USER= and PASSWORD=. △

The SAS/ACCESS Interface to Oracle Rdb uses the following statements in interactive line, noninteractive, or batch mode.

PROC DBLOAD <DBMS=RDB> <DATA=<*libref.*>*SAS-data-set*> <APPEND>;
ACCDESC=<*libref.access-descriptor*>;
COMMIT=*commit-frequency*;
DATABASE= <'>*Oracle-Rdb-pathname*<'>;
DELETE *variable-identifier-1*<*...variable-identifier-n*>;
ERRLIMIT=*error-limit*;
LABEL;
LIMIT=*load-limit*;
LIST <*list-selection*>;
LOAD;
NULLS *variable-identifier-1*=Y | N<*...variable-identifier-n*=Y | N>;
QUIT;
RENAME *variable-identifier-1*=<'>*column-name-1*<'>
<*...variable-identifier-n* =<'>*column-name-n*<'>>;
RESET ALL | *variable-identifier-1*<*...variable-identifier-n*>;
SQL *Oracle-Rdb SQL-statement*;
TABLE=<'>*Oracle-Rdb-table-name*<'>;
TYPE *variable-identifier-1* = <'>*column-type-1*'
<*...variable-identifier-n*='*column-type-n*'>;
WHERE *SAS-where-expression*;
RUN;

Statements

DATABASE=<'>*Oracle-Rdb-pathname*<'>;
indicates the name and physical location of the Oracle Rdb database where you want to create the new table. The name can be the operating-system-specific pathname or an OpenVMS logical name that points to the fully qualified Oracle Rdb database name. The .RDB extension is optional. If you specify a database, it must exist. If you do not know the names of your databases, contact your database administrator.

If you are accessing a remote database, you can specify the OpenVMS node name as part of the OpenVMS pathname of the database:

DATABASE=<'><*OpenVMS-net-node::*> *OpenVMS-pathname*<'>;

If DATABASE= is not specified, the default action is to use the value of the OpenVMS logical name SQL$DATABASE. For more information about SQL$DATABASE, see your Oracle Rdb documentation.

Note: Double quotation marks cannot be used with this option. △

The following example creates a new Oracle Rdb table, EXCHANGE, from the DLIB.RATEOFEX data file. An access descriptor ADLIB.EXCHANGE is also created, based on the new table. You must be granted the appropriate privileges to create new Oracle Rdb tables or views. This example also uses a DATABASE= statement for a remote database called TEXTILE on the OpenVMS node ATLANTA.

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

```
proc dbload dbms=rdb data=dlib.rateofex;
   database=atlanta::disk1:[root]textile.rdb;
   table=exchange;
   accdesc=adlib.exchange;
   rename fgnindol=fgnindoll 4=dollarsfgn;
   nulls updated=n fgnindoll=n 4=n country=n;
   load;
run;
```

Note: Rdb has a 30–character limit on its database column names. Therefore, when you load a SAS data set into Rdb, be sure SAS variable names are no longer than 30 characters. △

SQL Procedure Pass-Through Facility: Oracle Rdb Specifics

Chapter 6, "SQL Procedure's Interaction with SAS/ACCESS Software" describes the generic PROC SQL statements that you use to connect to and disconnect from a DBMS, to send DBMS-specific statements to the DBMS, and to retrieve DBMS data for your SAS programs. The following section describes the Oracle Rdb-specific arguments that you use in the CONNECT statement"Arguments to Connect to Oracle Rdb" on page 7.

Arguments to Connect to Oracle Rdb

CONNECT is an optional statement that can be used when connecting to Oracle Rdb. Oracle Rdb has one *database-connection-argument* that you can specify in this statement. It can also be used to connect to multiple Oracle Rdb systems.

CONNECT TO RDB <AS *alias*> <(DATABASE=<'>*Oracle-Rdb-pathname*<'>;>

(DATABASE=<'>*Oracle-Rdb-pathname*<'>)
specifies the name and physical location of the Oracle Rdb database. The name can be the operating-system-specific pathname or an OpenVMS logical name that points to the fully qualified Oracle Rdb database name. The entire DATABASE= argument must be enclosed in parentheses. The .RDB extension is optional.

If you are accessing a remote database, you can specify the OpenVMS node name as part of the OpenVMS pathname of the database:

(DATABASE=<'>*<Oracle-Rdb-net-node::> OpenVMS-pathname*<'>)

The DATABASE= argument is optional. If you specify a database, it must exist. If you do not know the names of your databases, contact your database administrator.

If the CONNECT statement or the DATABASE= argument is omitted, the default action is to use the value of the OpenVMS logical name SQL$DATABASE. For more information about SQL$DATABASE, see your Oracle Rdb documentation.

Note: Double quotation marks cannot be used with this option. △

The following example connects to Oracle Rdb and sends it two EXECUTE statements. The database name is TEXTILE, and it is located on an OpenVMS node named ATLANTA.

```
proc sql;
   connect to rdb
     (database=atlanta::disk1:[root]textile.rdb);
   execute (create view whotookord as
            select ordernum, takenby, firstname,
              lastname, phone
              from orders, employees
              where orders.takenby=employees.empid)
           by rdb;
   execute (grant select on whotookord to sasdemo)
     by rdb;
   disconnect from rdb;
quit;
```

Note: Rdb has a 30–character limit on its database column names. Therefore, when you create an Rdb table using the PROC SQL EXECUTE statement, be sure the column names are no longer than 30 characters. △

Requesting READONLY Access to an Oracle Rdb Table Using the Pass-Through Facility

The READONLY option is used in the CONNECT statement of the SQL Procedure Pass-Through Facility. It enables you to use a PROC SQL view without locking other users out of the table.

The option values are READONLY | NOREADONLY. The default value is NOREADONLY. This example illustrates the option:

```
connect to rdb(database=atlanta::disk1:[root]
textile.rdb
  readonly);
```

When you specify the READONLY option value, a SET TRANSACTION READ ONLY command is executed after the connection to the database is established. This option is useful if you want to create a permanent PROC SQL view that contains a Pass-Through query. This view can then be used without locking other users out of the table. For example, when you use FSBROWSE on the view, the data is read in a READ ONLY transaction. The following example connects to Oracle Rdb and creates a view using the READONLY option. The database name is TEXTILE, and is located on an OpenVMS node named ATLANTA.

```
proc sql;
   connect to rdb
     (database=atlanta::disk1:[root]textile.rdb
      readonly);
   create view invoice as
            select *
              from connection to
              (select * from invoice);
   disconnect from rdb;
quit;
```

When you create a PROC SQL view, any arguments that you specify in the corresponding CONNECT statement are stored also. Thus, when the PROC SQL view is used in a SAS program, the SAS System can establish the appropriate connection to the DBMS.

Oracle Rdb Naming Conventions

Oracle Rdb database objects that can be named include tables, views, columns, indexes. Use the following Oracle Rdb naming conventions:

- A name must start with a letter.
- A name can be from 1 to 30 characters long.
- A name can contain the letters A through Z, the digits 0 through 9, the underscore (_), and the dollar sign ($). (Although dollar signs are permitted in names, they are also used in some operating system-supplied names. To avoid conflicts with these names, you should not to use dollar signs in your table, view, column, or index names.)
- Names are not case sensitive; for example, CUSTOMER is the same as Customer.
- A name must end with a letter or a digit.
- A name must not be an Oracle Rdb reserved word.
- A name must not be the same as another Oracle Rdb database object that has the same type.

Oracle Rdb Data Types

Every column in a table has a name and a data type. The data type tells Oracle Rdb how much physical storage to set aside for the column and the format in which the data is stored. The Oracle Rdb data types are described in the following tables

For more information on these data types, see your Oracle Rdb documentation.

String Data

Table 1.1 String Data Types

Oracle Rdb SQL	Description
CHAR(*n*)	fixed-length column for character string data composed of 8-bit ASCII bytes where *n* specifies the length of the string. Use an unsigned integer (*n*). If you omit *n* , a 1-character column is created. Maximum *n* is 65,271.
VARCHAR(*n*)	varying-length column for character string data composed of 8-bit ASCII bytes, where *n* specifies the maximum length of the string. Use an unsigned integer (*n*). Maximum *n* is 65,269 characters.
LONG VARCHAR	varying-length column for character string data, which has 16,383 characters. LONG VARCHAR is equivalent to specifying VARCHAR(16,383).

Numeric Data

Table 1.2 Numeric Data Types

Oracle Rdb SQL	Description
TINYINT(*n*)*	a tiny integer (signed byte, 8 contiguous bits). Can store a range of values from –128 through 127.
SMALLINT(*n*)*	a small integer (signed 16-bit word). Can store a range of values from –32,768 to 32,767.
INTEGER(*n*)*	an integer (signed 32-bit longword). Can store a range of values from –2**31 to (2**31)–1. You can abbreviate INTEGER to INT in your program.
BIGINT(*n*)*	a signed 64-bit quadword. Can store a range of values from –2**63 to (2**63) –1.
FLOAT (*n*)	a real (32-bit) or double precision (64-bit) floating-point number, depending on the precision indicated in the positive integer (*n*). If *n* is less than 25, FLOAT specifies a 32-bit floating-point number. If *n* is 25 or greater, FLOAT specifies a 64-bit floating-point number. The maximum value for *n* is 53. If *n* is omitted, FLOAT specifies a 64-bit floating-point number.
REAL	a floating-point number (32-bit) with precision to 24 binary digits.
DOUBLE PRECISION	a floating-point number (64-bit) with precision to 53 binary digits. (The way this data type is stored is the closest match to the way that the SAS System stores its numeric data type; therefore, numeric columns of this type require the least processing when being accessed by the SAS System.)

* (*n*) represents a scale factor that indicates the number of places to the right of the decimal point, and must be an integer in the range from 0 to 127.

Date-Time Data

Table 1.3 Date-time Data Types

Oracle Rdb SQL	Description
DATE ANSI	specifies a DATE that contains YEAR to DAY
DATE |DATE VMS	specifies a timestamp that contains YEAR to SECOND

Oracle Rdb Nulls

Oracle Rdb has a special value that is called NULL, which signifies an absence of information and is analogous to the SAS System's *missing value*. By default, columns allow NULL values.

Columns can be defined so that they cannot contain NULL data. For example, the CREATE TABLE statement for the CUSTOMERS table defines the first column, CUSTOMER, as CHAR(8) and NOT NULL. NOT NULL tells Oracle Rdb not to add a row to the table unless the row has a value for CUSTOMER.

ACCESS Procedure Data Conversions

The following table shows the default SAS System variable formats that the ACCESS procedure assigns to each Oracle Rdb SQL data type. To calculate some of the SAS formats (for example, **(5+*n*).*n***), see the explanation following the table.

Table 1.4 Default SAS System Variable Formats for Oracle Rdb SQL Data Types—ACCESSProcedure

Oracle Rdb SQL Data Type	SAS Variable Format*
CHAR(*n*)	$*n*.*n* < 200
	$200. *n* >= 200
VARCHAR(*n*)	$*n*.*n* < 200
	$200. *n* >= 200
LONG VARCHAR	$200.
TINYINT	4.0
TINYINT(*n*)	(3+*n*).*n*
SMALLINT	6.0
SMALLINT(*n*)	(5 + *n*).*n*
INTEGER	11.0
INTEGER(*n*)	(10 + *n*).*n*
BIGINT	20.0
BIGINT(*n*)	(19 + *n*).*n*

Oracle Rdb SQL Data Type	SAS Variable Format*
REAL	E14.0
DOUBLE PRECISION	E23.0
DATE	DATETIME21.2
DECIMAL NUMERIC(*n*,*n*)	Oracle Rdb SQL converts to other numeric type. SAS/ACCESS supports the converted type.
LIST OF BYTE VARYING	unsupported
datetime intervals	unsupported

* *constant* + *n.n* in Oracle Rdb SQL data types is equivalent to *w.d* in SAS formats.

To determine how an Oracle Rdb SQL data type is formatted by the SAS System, use the following conversion example, SMALLINT(*n*) to (5 + *n*).*n* :

SMALLINT(1) is equivalent to a SAS format of 6.1, or (5 + 1).1

SMALLINT(2) is equivalent to a SAS format of 7.2, or (5 + 2).2

SMALLINT(5) is equivalent to a SAS format of 10.5, or (5 + 5).5

Thus, the value of *n* is added to the 5 and the value of *n* also replaces the decimal value.

If Oracle Rdb data falls outside valid SAS data ranges, you get an error message in the SAS log when you try to read the data. For example, an Oracle Rdb SQL date might not fall in the valid SAS date range.

DBLOAD Procedure Data Conversions

The following table shows the default Oracle Rdb data types that the DBLOAD procedure assigns to each SAS variable format.

Table 1.5 Default Oracle Rdb Data Types for SAS System Variable Formats—DBLOADProcedure

SAS Variable Format	Oracle Rdb SQL Data Type
IB*w.d*, PIB*w.d*	INTEGER
w.d	TINYINT(*d*)
w.d	SMALLINT(*d*)
w.d	INTEGER(*d*)
w.d	BIGINT(*d*) QUADWORD(*d*)
w.d	DOUBLE PRECISION
$*n*.	CHAR(*n*)

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ACCESS® Software for Relational Databases: Reference, Version 8 (Oracle Rdb® Chapter)*, Cary, NC: SAS Institute Inc., 1999.

SAS/ACCESS® Software for Relational Databases: Reference, Version 8 (Oracle Rdb® Chapter)

ISBN 1–58025–536–1

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

1st printing, October 1999