Oracle® Database Gateway for Teradata

User's Guide

11*g* Release 2 (11.2)

E12068-03

January 2012

ORACLE®

Oracle Database Gateway for Teradata User's Guide, 11*g* Release 2 (11.2)

E12068-03

Primary Author: Maitreyee Chaliha

Contributor: Vira Goorah, Juan Pablo Ahues-Vasquez, Peter Castro, Charles Benet, Peter Wong, and Govind Lakkoju

Contents

C Data Dictionary

D Initialization Parameters

List of Tables

Preface

This manual describes the Oracle Database Gateway for Teradata, which enables Oracle client applications to access Teradata data through Structured Query Language (SQL). The gateway, with the Oracle database, creates the appearance that all data resides on a local Oracle database, even though the data can be widely distributed.

This preface covers the following topics:

Audience

This manual is intended for Oracle database administrators who perform the following tasks:

- Installing and configuring the Oracle Database Gateway for Teradata
- Diagnosing gateway errors
- Using the gateway to access Teradata data

> **Note:** You should understand the fundamentals of Oracle Database Gateways and the UNIX based platforms before using this guide to install or administer the gateway.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see these Oracle resources:

- *Oracle Database New Features Guide*
- *Oracle Call Interface Programmer's Guide*
- *Oracle Database Administrator's Guide*
- *Oracle Database Advanced Application Developer's Guide*
- *Oracle Database Concepts*
- *Oracle Database Performance Tuning Guide*
- *Oracle Database Error Messages*
- *Oracle Database Globalization Support Guide*
- *Oracle Database Reference*
- *Oracle Database SQL Language Reference*
- *Oracle Database Net Services Administrator's Guide*
- *SQL*Plus User's Guide and Reference*
- *Oracle Database Heterogeneous Connectivity User's Guide*
- *Oracle Database Security Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction

This chapter introduces the challenge faced by organizations when running several different database systems. It briefly covers Heterogeneous Services, the technology that the Oracle Database Gateway for Teradata is based on.

To get a good understanding of generic gateway technology, Heterogeneous Services, and how Oracle Database Gateways fit in the picture, reading the *Oracle Database Heterogeneous Connectivity User's Guide* first is highly recommended.

This chapter contains the following sections:

Overview

Heterogeneous data access is a problem that affects a lot of companies. A lot of companies run several different database systems. Each of these systems stores data and has a set of applications that run against it. Consolidation of this data in one database system is often hard - in large part because many of the applications that run against one database may not have an equivalent that runs against another. Until such time as migration to one consolidated database system is made feasible, it is necessary for the various heterogeneous database systems to interoperate.

Oracle Database Gateways provide the ability to transparently access data residing in a non-Oracle system from an Oracle environment. This transparency eliminates the need for application developers to customize their applications to access data from different non-Oracle systems, thus decreasing development efforts and increasing the mobility of the application. Applications can be developed using a consistent Oracle interface for both Oracle and Teradata.

Gateway technology is composed of two parts: a component that has the generic technology to connect to a non-Oracle system, which is common to all the non-Oracle systems, called Heterogeneous Services, and a component that is specific to the non-Oracle system that the gateway connects to. Heterogeneous Services, in conjunction with the Oracle Database Gateway agent, enables transparent access to non-Oracle systems from an Oracle environment.

Heterogeneous Services Technology

Heterogeneous Services provides the generic technology for connecting to non-Oracle systems. As an integrated component of the database, Heterogeneous Services can

exploit features of the database, such as the powerful SQL parsing and distributed optimization capabilities.

Heterogeneous Services extend the Oracle SQL engine to recognize the SQL and procedural capabilities of the remote non-Oracle system and the mappings required to obtain necessary data dictionary information. Heterogeneous Services provides two types of translations: the ability to translate Oracle SQL into the proper dialect of the non-Oracle system as well as data dictionary translations that displays the metadata of the non-Oracle system in the local format. For situations where no translations are available, native SQL can be issued to the non-Oracle system using the pass-through feature of Heterogeneous Services.

Heterogeneous Services also maintains the transaction coordination between Oracle and the remote non-Oracle system, such as providing the two-phase commit protocol to ensure distributed transaction integrity, even for non-Oracle systems that do not natively support two-phase commit.

> **See Also:** *Oracle Database Heterogeneous Connectivity User's Guide* for more information about Heterogeneous Services.

Oracle Database Gateways

The capabilities, SQL mappings, data type conversions, and interface to the remote non-Oracle system are contained in the gateway. The gateway interacts with Heterogeneous Services to provide the transparent connectivity between Oracle and non-Oracle systems.

The gateway must be installed on a machine running either the Teradata database or the Teradata client. This machine can be the same machine as the Oracle database or on the same machine as the Teradata database or on a third machine as a standalone. Each configuration has its advantages and disadvantages. The issues to consider when determining where to install the gateway are network traffic, operating system platform availability, hardware resources and storage.

2

Teradata Gateway Features and Restrictions

After the gateway is installed and configured, you can use the gateway to access Teradata data, pass Teradata commands from applications to the Teradata database, perform distributed queries, and copy data.

This chapter contains the following sections:

Using the Pass-Through Feature

The gateway can pass Teradata commands or statements from the application directly to the Teradata database using the `DBMS_HS_PASSTHROUGH` package.

Use the `DBMS_HS_PASSTHROUGH` package in a PL/SQL block to specify the statement to be passed to the Teradata database, as follows:

```
DECLARE
    num_rows INTEGER;
BEGIN
    num_rows := DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@TERA('command');
END;
/
```

Where *command* cannot be one of the following:

- `BEGIN TRANSACTION`
- `BT`
- `COMMIT`
- `END TRANSACTION`
- `ET`

- ROLLBACK

The DBMS_HS_PASSTHROUGH package supports passing bind values and executing SELECT statements.

> **See Also:** *Oracle Database PL/SQL Packages and Types Reference* and Chapter 3, Features of Oracle Database Gateways, of *Oracle Database Heterogeneous Connectivity User's Guide* for more information about the DBMS_HS_PASSTHROUGH package.

Executing Stored Procedures and Functions

Using the procedural feature, the gateway can execute stored procedures that are defined in the Teradata database. It is not necessary to relink the gateway or define the procedure to the gateway, but the procedure's access privileges must permit access by the gateway.

> **See Also:** *Oracle Database Heterogeneous Connectivity User's Guide* for more information about executing stored procedures.

Standard PL/SQL statements are used to execute a stored procedure.

The gateway supports stored procedures in three mutually exclusive modes:

- Normal mode: Have access to IN/OUT arguments only
- Return value mode: Have a return value for all stored procedures
- Resultset mode: Out values are available as last result set

Return Values and Stored Procedures

By default, all stored procedures and functions do not return a return value to the user. To enable return values, set the HS_FDS_PROC_IS_FUNC parameter in the initialization parameter file.

> **See Also:** Appendix D, "Initialization Parameters" for information about both editing the initialization parameter file and the HS_FDS_PROC_IS_FUNC parameter.

Note: If you set the HS_FDS_PROC_IS_FUNC gateway initialization parameter, you must change the syntax of the procedure execute statement for all existing stored procedures.

In the following example, the employee name JOHN SMYTHE is passed to the Teradata stored procedure REVISE_SALARY. The stored procedure retrieves the salary value from the Teradata database to calculate a new yearly salary for JOHN SMYTHE. The revised salary returned in RESULT is used to update EMP in a table of an Oracle database:

```
DECLARE
  INPUT VARCHAR2(15);
  RESULT NUMBER(8,2);
BEGIN
  INPUT := 'JOHN SMYTHE';
  RESULT := REVISE_SALARY@TERA(INPUT);
  UPDATE EMP SET SAL = RESULT WHERE ENAME =: INPUT;
```

```
END;
/
```

The procedural feature automatically converts non-Oracle data types to and from PL/SQL data types.

Result Sets and Stored Procedures

The Oracle Database Gateway for Teradata provides support for stored procedures which return result sets.

By default, all stored procedures and functions do not return a result set to the user. To enable result sets, set the `HS_FDS_RESULTSET_SUPPORT` parameter in the initialization parameter file.

> **See Also:** Appendix D, "Initialization Parameters" for information about both editing the initialization parameter file and the `HS_FDS_RESULTSET_SUPPORT` parameter. For further information about Oracle support for result sets in non-Oracle databases see *Oracle Database Heterogeneous Connectivity User's Guide*.

Note: If you set the `HS_FDS_RESULTSET_SUPPORT` gateway initialization parameter, you must change the syntax of the procedure execute statement for all existing stored procedures or errors will occur.

When accessing stored procedures with result sets through the Oracle Database Gateway for Teradata, you will be in the sequential mode of Heterogeneous Services.

The Oracle Database Gateway for Teradata returns the following information to Heterogeneous Services during procedure description:

- All the input arguments of the remote stored procedure
- None of the output arguments
- One out argument of type ref cursor (corresponding to the first result set returned by the stored procedure)

Client programs have to use the virtual package function `dbms_hs_result_set.get_next_result_set` to get the ref cursor for subsequent result sets. The last result set returned is the out argument from the procedure.

The limitations of accessing result sets are the following:

- Result sets returned by a remote stored procedure have to be retrieved in the order in which they were placed on the wire
- On execution of a stored procedure, all result sets returned by a previously executed stored procedure will be closed (regardless of whether the data has been completely

In the following example, the Teradata stored procedure is executed to fetch the contents of the `emp` and `dept` tables from Teradata:

```
CREATE PROCEDURE refcurproc (@arg1 varchar(255), @arg2 varchar(255) output)
AS
SECLECT @arg2 = @arg1
SELECT * FROM EMP
SELECT * FROM DEPT
```

```
GO
```

This stored procedure assigns the input parameter `arg1` to the output parameter `arg2`, opens the query `SELECT * FROM EMP` in ref cursor `rc1`, and opens the query `SELECT * FROM DEPT` in ref cursor `rc2`.

> **Note:** Chained mode must be set before creating the stored procedure. Issue the following command in Teradata: `set chained on`

OCI Program Fetching from Result Sets in Sequential Mode

The following example shows OCI program fetching from result sets in sequential mode:

```
OCIEnv *ENVH;
OCISvcCtx *SVCH;
OCIStmt *STMH;
OCIError *ERRH;
OCIBind *BNDH[3];
OraText arg1[20];
OraText arg2[255];
OCIResult *rset;
OCIStmt *rstmt;
ub2 rcode[3];
ub2 rlens[3];
sb2 inds[3];
OraText *stmt = (OraText *) "begin refcurproc@TERA(:1,:2,:3); end;";
OraText *n_rs_stm = (OraText *)
  "begin :ret := DBMS_HS_RESULT_SET.GET_NEXT_RESULT_SET@TERA; end;";

/* Prepare procedure call statement */

/* Handle Initialization code skipped */
OCIStmtPrepare(STMH, ERRH, stmt, strlen(stmt), OCI_NTV_SYNTAX, OCI_DEFAULT);

/* Bind procedure arguments */
inds[0] = 0;
strcpy((char *) arg1, "Hello World");
rlens[0] = strlen(arg1);
OCIBindByPos(STMH, &BNDH[0], ERRH, 1, (dvoid *) arg1, 20, SQLT_CHR,
             (dvoid *) &(inds[0]), &(rlens[0]), &(rcode[0]), 0, (ub4 *) 0,
             OCI_DEFAULT);
inds[1] = -1;
OCIBindByPos(STMH, &BNDH[1], ERRH, 1, (dvoid *) arg2, 20, SQLT_CHR,
             (dvoid *) &(inds[1]), &(rlens[1]), &(rcode[1]), 0, (ub4 *) 0,
             OCI_DEFAULT);

inds[2] = 0;
rlens[2] = 0;
OCIDescriptorAlloc(ENVH, (dvoid **) &rset, OCI_DTYPE_RSET, 0, (dvoid **) 0);
OCIBindByPos(STMH, &BNDH[2], ERRH, 2, (dvoid *) rset, 0, SQLT_RSET,
             (dvoid *) &(inds[2]), &(rlens[2]), &(rcode[2]),
             0, (ub4 *) 0, OCI_DEFAULT);

/* Execute procedure */
OCIStmtExecute(SVCH, STMH, ERRH, 1, 0, (CONST OCISnapshot *) 0,
               (OCISnapshot *) 0, OCI_DEFAULT);
```

```
/* Convert result set to statement handle */
OCIResultSetToStmt(rset, ERRH);
rstmt = (OCIStmt *) rset;

/* After this the user can fetch from rstmt */
/* Issue get_next_result_set call to get handle to next_result set */
/* Prepare Get next result set procedure call */

OCIStmtPrepare(STMH, ERRH, n_rs_stm, strlen(n_rs_stm), OCI_NTV_SYNTAX,
               OCI_DEFAULT);

/* Bind return value */
OCIBindByPos(STMH, &BNDH[1], ERRH, 1, (dvoid *) rset, 0, SQLT_RSET,
             (dvoid *) &(inds[1]), &(rlens[1]), &(rcode[1]),
             0, (ub4 *) 0, OCI_DEFAULT);

/* Execute statement to get next result set*/
OCIStmtExecute(SVCH, STMH, ERRH, 1, 0, (CONST OCISnapshot *) 0,
               (OCISnapshot *) 0, OCI_DEFAULT);

/* Convert next result set to statement handle */
OCIResultSetToStmt(rset, ERRH);
rstmt = (OCIStmt *) rset;

/* Now rstmt will point to the second result set returned by the
remote stored procedure */

/* Repeat execution of get_next_result_set to get the output arguments */
```

PL/SQL Program Fetching from Result Sets in Sequential Mode

Assume that the table `loc_emp` is a local table exactly like the Teradata emp table. The same assumption applies for `loc_dept`. `outargs` is a table with columns corresponding to the out arguments of the Teradata stored procedure.

```
create or replace package rcpackage is
  type RCTYPE is ref cursor;
end rcpackage;
/
declare
  rc1 rcpackage.rctype;
  rec1 loc_emp%rowtype;
  rc2 rcpackage.rctype;
  rec2 loc_dept%rowtype;
  rc3 rcpackage.rctype;
  rec3 outargs%rowtype;
  out_arg varchar2(255);

begin

  -- Execute procedure
  out_arg := null;
  refcurproc@TERA('Hello World', out_arg, rc1);

  -- Fetch 20 rows from the remote emp table and insert them into loc_emp
  for i in 1 .. 20 loop
    fetch rc1 into rec1;
    insert into loc_emp (rec1.empno, rec1.ename, rec1.job,
    rec1.mgr, rec1.hiredate, rec1.sal, rec1.comm, rec1.deptno);
  end loop;
```

```
  -- Close ref cursor
  close rc1;

  -- Get the next result set returned by the stored procedure
  rc2 := dbms_hs_result_set.get_next_result_set@TERA;

  -- Fetch 5 rows from the remote dept table and insert them into loc_dept
  for i in 1 .. 5 loop
    fetch rc2 into rec2;
    insert into loc_dept values (rec2.deptno, rec2.dname, rec2.loc);
  end loop;

  --Close ref cursor
  close rc2;

  -- Get the output arguments from the remote stored procedure
  -- Since we are in sequential mode, they will be returned in the
  -- form of a result set
  rc3 := dbms_hs_result_set.get_next_result_set@TERA;

  -- Fetch them and insert them into the outarguments table
  fetch rc3 into rec3;
  insert into outargs (rec3.outarg, rec3.retval);

  -- Close ref cursor
  close rc3;

end;
/
```

CHAR Semantics

This feature allows the gateway to optionally run in CHAR Semantics mode. Rather than always describing Teradata CHAR columns as CHAR(n BYTE), this feature describes them as CHAR(n CHAR) and VARCHAR(n CHAR). The concept is similar to Oracle database CHAR Semantics. You need to specify HS_NLS_LENGTH_SEMANTICS=CHAR to activate this option. Refer to Appendix D for more detail.

Multi-byte Character Sets Ratio Suppression

This feature optionally suppresses the ratio expansion from Teradata database to Oracle databases involving a multi-byte character set (for example, from US7ASCII to AL32UTF8, or from KO16MSWIN949 to KO16KSC5601). By default, Oracle gateways assume the worst ratio to prevent data being truncated or allocation of buffer with insufficient size. However, if you have specific knowledge of your Teradata database and do not want the expansion to occur, you can specify HS_KEEP_REMOTE_COLUMN_SIZE parameter to suppress the expansion. Refer to Appendix D for more detail.

IPv6 Support

Besides full IPv6 support between Oracle databases and the gateway, IPv6 is also supported between this gateway and Teradata. Refer to HS_FDS_CONNECT_INFO parameter in Appendix D for more detail.

Gateway Session IDLE Timeout

You can optionally choose to terminate long idle gateway sessions automatically with the gateway parameter `HS_IDLE_TIMEOUT`. Specifically, when a gateway session is idle for more than the specified time limit, the gateway session is terminated with any pending changes rolled back.

Database Compatibility Issues for Teradata

Teradata and Oracle databases function differently in some areas, causing compatibility problems. The following compatibility issues are described in this section:

- Schema Considerations
- Naming Rules
- Data Types
- Queries
- Locking

Schema Considerations

The Oracle concept of a schema does not exist in Teradata. A schema name included in a query is recognized by Teradata as a database name. In the Oracle database, the schema of an object is identical to the owner of that object. However, when one retrieves or references an `OWNER` field in a Data Dictionary view such as `ALL_TABLES`, the `OWNER` field really is referencing a Teradata database name.

When querying data dictionary tables, the following results are returned:

- `ALL_*` data dictionary tables, data for every Teradata database is returned. The Teradata database name is returned in the `OWNER`, `INDEX_OWNER`, or `TABLE_OWNER` column depending on the data dictionary table being referenced.
- `ALL_USERS` data dictionary table, each Teradata database name is returned in the `USERNAME` column.
- `USER_*` data dictionary tables, data for the default Teradata database is returned for the `OWNER` or `TABLE_OWNER` column depending on the data dictionary table being referenced. If a default Teradata database is not defined, the DBC Teradata system database is used.

Naming Rules

Naming rule issues include the following:

- Rules for Naming Objects
- Case Sensitivity

Rules for Naming Objects

Oracle and Teradata use different database object naming rules. For example, the maximum number of characters allowed for each object name can be different. Also, the use of single and double quotation marks, case sensitivity, and the use of alphanumeric characters can all be different.

See Also: *Oracle Database Reference* and Teradata documentation.

Case Sensitivity

The Oracle database defaults to uppercase unless you surround identifiers with double quote characters. For example, to refer to the Teradata table called `emp`, enter the name with double quote characters, as follows:

```
SQL> SELECT * FROM "emp"@TERA;
```

However, to refer to the Teradata table called `emp` in the Teradata database named Scott from an Oracle application, enter the following:

```
SQL> SELECT * FROM "Scott"."emp"@TERA;
```

If the Teradata table called `emp` in the Teradata database named `SCOTT`, a name consisting entirely of uppercase letters, you can enter the owner name without double quote characters, as follows:

```
SQL> SELECT * FROM SCOTT."emp"@TERA;
```

or

```
SQL> SELECT * FROM scott."emp"@TERA;
```

Oracle recommends that you surround all Teradata object names with double quote characters and use the exact letter case for the object names as they appear in the Teradata data dictionary. This convention is not required when referring to the supported Oracle data dictionary tables or views listed in Appendix C, "Data Dictionary".

If existing applications cannot be changed according to these conventions, create views in Oracle to associate Teradata names to the correct letter case. For example, to refer to the Teradata table `emp` from an existing Oracle application by using only uppercase names, define the following view:

```
SQL> CREATE VIEW EMP (EMPNO, ENAME, SAL, HIREDATE)
      AS SELECT "empno", "ename", "sal", "hiredate"
      FROM "emp"@TERA;
```

With this view, the application can issue statements such as the following:

```
SQL> SELECT EMPNO, ENAME FROM EMP;
```

Using views is a workaround solution that duplicates data dictionary information originating in the Teradata data dictionary. You must be prepared to update the Oracle view definitions whenever the data definitions for the corresponding tables are changed in the Teradata database.

Data Types

Data type issues include the following:

- Binary Literal Notation
- Data Type Conversion

Binary Literal Notation

Oracle SQL uses hexadecimal digits surrounded by single quotes to express literal values being compared or inserted into columns defined as data type `RAW`.

This notation is not converted to syntax compatible with the Teradata `VARBINARY` and `BINARY` data types (a `ff` surrounded by single quotes followed by hexadecimal digits).

For example, the following statement is not supported:

```
SQL> INSERT INTO BINARY_TAB@TERA VALUES ('ff'xb)
```

where BINARY_TAB contains a column of data type VARBINARY or BINARY. Use bind variables when inserting into or updating VARBINARY and BINARY data types.

Data Type Conversion

Teradata does not support implicit date conversions. Such conversions must be explicit.

For example, the gateway issues an error for the following SELECT statement:

```
SELECT DATE_COL FROM TEST@TERA WHERE DATE_COL = "1-JAN-2001";
```

To avoid problems with implicit conversions, add explicit conversions, as in the following:

```
SELECT DATE_COL FROM TEST@TERA WHERE DATE_COL = TO_DATE("1-JAN-2001")
```

> **See Also:** Appendix A, "Data Type Conversion" for more information about restrictions on data types.

Queries

Query issues include the following:

- Row Selection
- Empty Bind Variables

Row Selection

Teradata evaluates a query condition for all selected rows before returning any of the rows. If there is an error in the evaluation process for one or more rows, no rows are returned even though the remaining rows satisfy the condition.

Oracle evaluates the query condition row-by-row and returns a row when the evaluation is successful. Rows are returned until a row fails the evaluation.

Empty Bind Variables

For VARCHAR bind variables, the gateway passes empty bind variables to the Teradata database as a NULL value.

Locking

The locking model for an Teradata database differs significantly from the Oracle model. The gateway depends on the underlying Teradata behavior, so the following possible scenarios can affect Oracle applications that access Teradata through the gateway:

- Read access might block write access.
- Write access might block read access.
- Statement-level read consistency is not guaranteed.

> **See Also:** Teradata documentation for information about the Teradata locking model.

Known Restrictions

If you encounter incompatibility problems not listed in this section or in "Known Problems" on page 2-12, contact Oracle Support Services. The following section describes the known restrictions and includes suggestions for dealing with them when possible:

Transactional Integrity

The gateway cannot guarantee transactional integrity in the following cases:

- When a statement that is processed by the gateway causes an implicit commit in the target database
- When the target database is configured to work in autocommit mode

---

Note: Oracle strongly recommends the following:

- If you know that executing a particular statement causes an implicit commit in the target database, then ensure that this statement is executed in its own transaction.
- Do not configure the target database to work in autocommit mode.

Transaction Capability

The gateway does not support savepoints. If a distributed update transaction is under way involving the gateway and a user attempts to create a savepoint, the following error occurs:

```
ORA-02070: database dblink does not support savepoint in this context
```

By default, the gateway is configured as `COMMIT_CONFIRM` and it is always the commit point site when the Teradata database is updated by the transaction.

COMMIT or ROLLBACK in PL/SQL Cursor Loops Closes Open Cursors

Any `COMMIT` or `ROLLBACK` issued in a PL/SQL cursor loop closes all open cursors, which can result in the following error:

```
ORA-1002:  fetch out of sequence
```

To prevent this error, move the `COMMIT` or `ROLLBACK` statement outside the cursor loop.

Pass-Through Feature

Oracle recommends that you place a DDL statement in its own transaction when executing such a statement with the pass-through feature. An explicit `COMMIT` must be issued after the DDL statement.

If the SQL statements being passed through the gateway result in an implicit commit at the Teradata database, the Oracle transaction manager is unaware of the commit and an Oracle `ROLLBACK` command cannot be used to roll back the transaction.

Bind Variables for Date Columns

You cannot compare columns of data type `TIME` or `TIMESTAMP` to a bind variable.

The following SQL statement causes an error message:

```
SQL> select time_column from time_table@TERA where time_column = :a;
```

The following error is issued:

```
Invalid operation on an ANSI Datetime or Interval value.
```

SQL Syntax

This section lists restrictions on the following SQL syntax:

- WHERE CURRENT OF Clause
- CONNECT BY Clause
- ROWID
- EXPLAIN PLAN Statement
- Callback Support

> **See Also:** Appendix B, "Supported SQL Syntax and Functions" for more information about restrictions on SQL syntax.

WHERE CURRENT OF Clause

`UPDATE` and `DELETE` statements with the `WHERE CURRENT OF` clause are not supported by the gateway because they rely on the Oracle `ROWID` implementation. To update or delete a specific row through the gateway, a condition style `WHERE` clause must be used.

CONNECT BY Clause

The gateway does not support the `CONNECT BY` clause in a `SELECT` statement.

ROWID

The Oracle `ROWID` implementation is not supported.

EXPLAIN PLAN Statement

The `EXPLAIN PLAN` statement is not supported.

Callback Support

SQL statements that require the gateway to callback to Oracle database are not supported.

The following categories of SQL statements will result in a callback:

- Any DML with a sub-select, which refers to a table in Oracle database. For example:

```
INSERT INTO emp@TERA SELECT * FROM oracle_emp;
```

- Any DELETE, INSERT, UPDATE or "SELECT... FOR UPDATE..." SQL statement containing SQL functions or statements that needs to be executed at the originating Oracle database.

  These SQL functions include USER, USERENV, and SYSDATE; and invovle the selection of data from the originating Oracle database. For example:

```
DELETE FROM emp@TERA WHERE hiredate > SYSDATE;

SELECT ename FROM tkhoemp@TERA
WHERE hiredate IN (SELECT hiredate FROM oracle_emp)
FOR UPDATE OF empno;
```

- Any SQL statement that involves a table in Oracle database, and a LONG or LOB column in a remote table. For example:

```
SELECT a.long1, b.empno FROM scott.table@TERA a, emp b
WHERE a.id=b.empno;

SELECT a.long1, b.dummy FROM table_non@TERA a, dual b;
```

where a.long1 is a LONG column.

SQL*Plus

You need to use double quotes to wrap around lowercase table names.

For example:

```
copy from tkhouser/tkhouser@inst1 insert loc_tkhodept using select * from
"tkhodept"@TERA;
```

Database Links

The gateway is not a shared server process and cannot support shared database links. Each gateway session spawns a separate gateway process and connections cannot be shared.

CALLBACK links

Oracle Database Gateway for Teradata does not support CALLBACK links. Attempting to use a CALLBACK link with the gateway will return the following error message:

```
ORA-02025: All tables in the SQL statement must be at the remote database
```

Known Problems

This section describes known problems and includes suggestions for correcting them when possible. If you have any questions or concerns about the problems, contact Oracle Support Services. A current list of problems is available online. Contact your local Oracle office for information about accessing the list.

The following known problems are described in this section:

- Teradata LONG VARCHAR Data Type

Teradata LONG VARCHAR Data Type

The following restrictions apply when using `LONG VARCHAR` data types:

- An unsupported SQL function cannot be used in a SQL statement that accesses a column defined as Teradata data type `LONG VARCHAR`.
- You cannot use SQL*Plus to select data from a column defined as Teradata data type `LONG VARCHAR` when the data is greater than 80 characters in length. Oracle recommends using Pro*C or Oracle Call Interface to access such data in a Teradata database.
- `LONG VARCHAR` data types must be `NULLABLE` for `INSERT` or `UPDATE` to work.
- A table including a `LONG VARCHAR` column must have a unique index defined on the table or the table must have a separate column that serves as a primary key.
- `LONG VARCHAR` data cannot be read through pass-through queries.

The gateway does not support the PL/SQL function `COLUMN_VALUE_LONG` of the `DBMS_SQL` package.

> **See Also:** Appendix B, "Supported SQL Syntax and Functions" for more information about restrictions on SQL syntax.

Schema Names and PL/SQL

If you do not prefix a Teradata database object with its schema name in a SQL statement within a PL/SQL block, the following error message occurs:

```
ORA-6550 PLS-201 Identifier table_name must be declared.
```

Change the SQL statement to include the schema name of the object.

Data Dictionary Views and PL/SQL

You cannot refer to data dictionary views in SQL statements that are inside a PL/SQL block.

3

Case Studies

The following case studies for Teradata demonstrate some of the features of the Oracle Database Gateway. You can verify that the gateway is installed and operating correctly by using the demonstration files included on the distribution media.

The demonstration files are automatically copied to disk when the gateway is installed.

This chapter contains the following sections:

Case Descriptions

The cases illustrate:

- A simple query (Case 1)
- A more complex query (Case 2)
- Joining Teradata tables (Case 3)
- Write capabilities (Case 4)
- A data dictionary query (Case 5)
- The pass-through feature (Case 6)

Installation Media Contents

The distribution media contains the following:

- Demonstration files.
- One SQL script file that creates the demonstration tables in the Teradata database.
- One SQL script file that drops the demonstration tables from the Teradata database.

Demonstration Files

After a successful gateway installation, use the demonstration files stored in the directory `$ORACLE_HOME/dg4tera/demo` where `$ORACLE_HOME` is the directory under which the gateway is installed. The directory contains the following demonstration files:

- `bldtera.sql`
- `case1.sql`
- `case2.sql`
- `case3.sql`
- `case4a.sql`
- `case4b.sql`
- `case4c.sql`
- `case5.sql`
- `case6a.sql`
- `case6b.sql`
- `droptera.sql`

Demonstration Requirements

The case studies assume these requirements have been met:

- The gateway demonstration tables are installed in the Teradata database.
- The Oracle database has an account named `SCOTT` with a password of `TIGER`.
- The Oracle database has a database link called `GTWLINK` (set up as public or private to the user `SCOTT`) that connects the gateway to a Teradata database as `SCOTT` with password `TIGER2`.

  For example, you can create the database link as follows:

  ```
  SQL> CREATE DATABASE LINK GTWLINK CONNECT TO SCOTT
    2    IDENTIFIED BY TIGER2 USING 'GTWSID';
  ```

- Oracle Net Services is configured correctly and running.

Creating Demonstration Tables

The case studies are based on the `GTW_EMP`, `GTW_DEPT`, and `GTW_SALGRADE` tables. If the demonstration tables have not been created in the Teradata database, use the `bldtera.sql` script to create them.

The script creates the demonstration tables in the Teradata database accordingly:

```
CREATE TABLE GTW_EMP (
EMPNO      SMALLINT NOT NULL
```

```
ENAME      VARCHAR(10),
JOB        VARCHAR(9),
MGR        SMALLINT,
HIREDATE   DATETIME,
SAL        NUMERIC(7,2),
COMM       NUMERIC(7,2),
DEPTNO     SMALLINT)
CREATE TABLE GTW_DEPT (
DEPTNO     SMALLINT NOT NULL,
DNAME      VARCHAR(14),
LOC        VARCHAR(13))
CREATE TABLE GTW_SALGRADE (
GRADE      REAL,
LOSAL      NUMERIC(9,4),
HISAL      NUMERIC(9,4))
```

Demonstration Table Definitions

The table definitions are listed here using information retrieved by the SQL*PLUS `DESCRIBE` command:

GTW_EMP

```
Name                           Null?    Type
------------------------------ -------- ----
EMPNO                          NOT NULL NUMBER(5)
ENAME                                   VARCHAR2(10)
JOB                                     VARCHAR2(9)
MGR                                     NUMBER(5)
HIREDATE                                DATE
SAL                                     NUMBER(7,2)
COMM                                    NUMBER(7,2)
DEPTNO                                  NUMBER(5)
```

GTW_DEPT

```
Name                           Null?    Type
------------------------------ -------- ----
DEPTNO                         NOT NULL NUMBER(5)
DNAME                                   VARCHAR2(14)
LOC                                     VARCHAR2(13)
```

GTW_SALGRADE

```
Name                           Null?    Type
------------------------------ -------- ----
GRADE                                   FLOAT(53)
LOSAL                                   NUMBER(9,4)
HISAL                                   NUMBER(9,4)
```

Demonstration Table Contents

The contents of the Teradata tables are:

GTW_EMP

```
EMPNO ENAME   JOB         MGR   HIREDATE   SAL   COMM   DEPTNO
----- -----   ---         ---   --------   ---   ----   ------
7369  SMITH   CLERK       7902  17-DEC-80   800             20
7499  ALLEN   SALESMAN    7698  20-FEB-81  1600   300       30
```

```
7521  WARD    SALESMAN    7698  22-FEB-81  1250   500       30
7566  JONES   MANAGER     7839  02-APR-81  2975              20
7654  MARTIN  SALESMAN    7698  28-SEP-81  1250  1400       30
7698  BLAKE   MANAGER     7839  01-MAY-81  2850              30
7782  CLARK   MANAGER     7839  09-JUN-81  2450              10
7788  SCOTT   ANALYST     7566  09-DEC-82  3000              20
7839  KING    PRESIDENT         17-NOV-81  5000              10
7844  TURNER  SALESMAN    7698  08-SEP-81  1500     0        30
7876  ADAMS   CLERK       7788  12-JAN-83  1100              20
7900  JAMES   CLERK       7698  03-DEC-81   950              30
7902  FORD    ANALYST     7566  03-DEC-81  3000              20
7934  MILLER  CLERK       7782  23-JAN-82  1300              10
```

GTW_DEPT

```
DEPTNO DNAME          LOC
------ -------------- --------
   10 ACCOUNTING     NEW YORK
   20 RESEARCH       DALLAS
   30 SALES          CHICAGO
   40 OPERATIONS     BOSTON
```

GTW_SALGRADE

```
GRADE      LOSAL      HISAL
------     ------     -----
    1        700      1200
    2       1201      1400
    3       1401      2000
    4       2001      3000
    5       3001      9999
```

Case 1: Simple Queries

Case 1 demonstrates the following:

- A simple query.
- A simple query retrieving full date information.

The first query retrieves all the data from `GTW_DEPT` and confirms that the gateway is working correctly.

Case 2: A More Complex Query

Case 2 demonstrates the following:

- The functions `SUM(`*`expression`*`)` and `NVL(`*`expr1, expr2`*`)` in the `SELECT` list.
- The `GROUP BY` and `HAVING` clauses.

This query retrieves the departments from `GTW_EMP` whose total monthly expenses are higher than `$10,000`.

Case 3: Joining Teradata Tables

Case 3 demonstrates the following:

- Joins between Teradata tables.
- Subselects.

The query retrieves information from three Teradata tables and relates the employees to their department name and salary grade, but only for those employees earning more than the average salary.

Case 4: Write Capabilities

Case 4 is split into three cases and demonstrates the following:

- DELETE Statement
- UPDATE Statement
- INSERT Statement

DELETE Statement

Case 4a demonstrates bind values and subselect. All employees in department `20` and one employee, `WARD`, in department `30` are deleted.

UPDATE Statement

Case 4b provides an example of a simple `UPDATE` statement. In this example, employees are given a `$100` a month salary increase.

INSERT Statement

Case 4c is an example of a simple insert statement that does not provide information for all columns.

Case 5: Data Dictionary Query

Case 5 demonstrates data dictionary mapping. It retrieves all the tables and views that exist in the Teradata database that begin with "`GTW`".

Case 6: The Pass-Through Feature

Case 6 demonstrates the gateway pass-through feature that allows an application to send commands or statements to Teradata.

This case demonstrates:

- A pass-through `UPDATE` statement using bind variables.
- A pass-through `SELECT` statement.

UPDATE Statement

Case 6a provides an example of a pass-through `UPDATE` statement with bind variables. In this example, the salary for `EMPNO 7934` is set to `4000`.

SELECT Statement

Case 6b provides an example of a pass-through `SELECT` statement. The data that is returned from the `SELECT` statement is inserted into a local table at the Oracle database.

A

Data Type Conversion

This appendix contains the following section:

- Data Type Conversion

Data Type Conversion

The gateway converts Teradata data types to Oracle data types as follows:

Table A–1 Data Type Conversions

Teradata	Oracle	Comment
BLOB	LONG RAW	
BYTE	RAW	-
BYTEINT	NUMBER(3)	BYTEINT range is -128 to +127
CHAR	CHAR	-
CHAR CHARACTER SET UNICODE	CHAR	-
CLOB	LONG	
DATE	CHAR(10)	-
DECIMAL	NUMBER(p[,s])	-
FLOAT	FLOAT(24)	with precision <= single precision float
FLOAT	FLOAT(53)	with precision > single precision float
INTEGER	NUMBER(10)	NUMBER range is -2,147,483,648 to 2,147,483,647
INTERVAL	CHAR	-
LONG VARCHAR CHARACTER SET UNICODE	LONG	if Oracle DB Character Set = Unicode. Otherwise, it is not supported.
LONG VARCHAR	LONG	if Oracle DB Character Set = Unicode. Otherwise, it is not supported
SMALLINT	NUMBER(5)	NUMBER range is -32768 to 32767
TIME	CHAR(15)	-
TIMESTAMP	DATE	-
VARBYTE	RAW	with less or equal to 2000 bytes
VARBYTE	LONG RAW	with more than 2000 bytes

Table A–1 (Cont.) Data Type Conversions

Teradata	Oracle	Comment
VARCHAR	VARCHAR2	If the size is greater than 4000, it is mapped to LONG
VARCHAR() CHARACTER SET UNICO	NVARCHAR2	If the size is 2000 or less, otherwise it is mapped to LONG

B

Supported SQL Syntax and Functions

This appendix contains the following sections:

Supported SQL Statements

With a few exceptions, the gateway provides full support for Oracle `DELETE`, `INSERT`, `SELECT`, and `UPDATE` statements.

The gateway does not support Oracle data definition language (DDL) statements. No form of the Oracle `ALTER`, `CREATE`, `DROP`, `GRANT`, or `TRUNCATE` statements can be used. Instead, use the pass-through feature of the gateway if you need to use DDL statements against the Teradata database.

> **See Also:** *Oracle Database SQL Language Reference* for detailed descriptions of keywords, parameters, and options.

DELETE

The `DELETE` statement is fully supported. However, only Oracle functions supported by Teradata can be used.

> **See Also:** "Functions Supported by Teradata" on page B-2 for a list of supported functions.

INSERT

The `INSERT` statement is fully supported. However, only Oracle functions supported by Teradata can be used.

> **See Also:** "Functions Supported by Teradata" on page B-2 for a list of supported functions.

SELECT

The `SELECT` statement is fully supported, with these exceptions:

- `CONNECT BY` *`condition`*
- `NOWAIT`
- `START WITH` *`condition`*
- `WHERE CURRENT OF`

UPDATE

The UPDATE statement is fully supported. However, only Oracle functions supported by Teradata can be used. Also, you cannot have SQL statements in the subquery that refer to the same table name in the outer query. Subqueries are not supported in the SET clause.

> **See Also:** "Functions Supported by Teradata" on page B-2 for a list of supported functions.

Oracle Functions

All functions are evaluated by the Teradata database after the gateway has converted them to Teradata SQL.

Functions Not Supported by Teradata

Oracle SQL functions with no equivalent function in Teradata are not supported in DELETE, INSERT, or UPDATE statements, but are evaluated by the Oracle database if the statement is a SELECT statement. That is, the Oracle database performs post-processing of SELECT statements sent to the gateway.

If an unsupported function is used in a DELETE, INSERT, or UPDATE, statement, the following Oracle error occurs:

```
ORA-02070: database db_link_name does not support function in this context
```

Functions Supported by Teradata

The gateway translates the following Oracle database functions in SQL statements to their equivalent Teradata functions:

- Arithmetic Operators
- Comparison Operators
- Group Functions
- String Functions
- Other Functions

Arithmetic Operators

Oracle	Teradata
+	+
-	-
*	*
/	/

Comparison Operators

Oracle	Teradata
=	=
>	>

Oracle	Teradata
<	<
>=	>=
<=	<=
<>, !=, ^=	<>, !=
IS NOT NULL	IS NOT NULL
IS NULL	IS NULL

Group Functions

Oracle	Teradata
AVG	AVG
COUNT	COUNT
MAX	MAX
MIN	MIN
SUM	SUM

String Functions

Oracle	Teradata
INSTR	POSITION

Other Functions

Oracle	Teradata
ABS	ABS
EXP	EXP
LOG(a, b)	LOG10(b) / LOG10(a)
POWER	**
SQRT	SQRT

C

Data Dictionary

The Oracle Database Gateway for Teradata User's Guide translates a query that refers to an Oracle database data dictionary table into a query that retrieves the data from Teradata system catalog tables. You perform queries on data dictionary tables over the database link in the same way you query data dictionary tables in the Oracle database. The gateway data dictionary is similar to the Oracle database data dictionary in appearance and use.

This appendix contains the following sections:

Data Dictionary Support

The following paragraphs describe the Oracle Database Gateway for Teradata data dictionary support.

Teradata System Catalog Tables

Teradata data dictionary information is stored in the Teradata database as Teradata system catalog tables. The Teradata system catalog tables define the structure of a database. When you change data definitions, Teradata reads and modifies the Teradata system catalog tables to add information about the user tables.

Accessing the Gateway Data Dictionary

Accessing a gateway data dictionary table or view is identical to accessing a data dictionary in an Oracle database. You issue a SQL `SELECT` statement specifying a database link. The Oracle database data dictionary view and column names are used to access the gateway data dictionary in an Oracle database. Synonyms of supported views are also acceptable. For example, the following statement queries the data dictionary table `ALL_CATALOG` to retrieve all table names in the Teradata database:

```
SQL> SELECT * FROM ALL_CATALOG@TERA;
```

When a data dictionary access query is issued, the gateway:

1. Maps the requested table, view, or synonym to one or more Teradata system table names. The gateway translates all data dictionary column names to their corresponding Teradata database column names within the query. Refer to "Data Dictionary Mapping" on page C-3 for details.

2. Sends the translated query to Teradata.
3. Might convert the retrieved Teradata data to give it the appearance of the Oracle database data dictionary table.
4. Passes the data dictionary information from the translated Teradata system catalog table to the Oracle database.

> **Note:** The values returned when querying the gateway data dictionary might not be the same as the ones returned by the Oracle Enterprise Manager `DESCRIBE` command.

Direct Queries to Teradata Tables

Queries issued directly to individual Teradata system catalog tables are allowed but they return different results because the Teradata system catalog table column names differ from those of the data dictionary view. Also, certain columns in a Teradata system catalog table cannot be used in data dictionary processing.

Supported Views and Tables

The gateway supports the following views and tables:

Supported Views and Tables	Supported Views and Tables
ALL_CATALOG	ALL_COL_COMMENTS
ALL_CONS_COLUMNS	ALL_CONSTRAINTS
ALL_IND_COLUMNS	ALL_INDEXES
ALL_OBJECTS	ALL_TAB_COLUMNS
ALL_TAB_COMMENTS	ALL_TABLES
ALL_USERS	ALL_VIEWS
DBA_CATALOG	DBA_COL_COMMENTS
DBA_OBJECTS	DBA_TABLES
DBA_TAB_COLUMNS	DBA_TAB_COMMENTS
DICT_COLUMNS	DICTIONARY
DUAL	USER_COL_COMMENTS
USER_CATALOG	USER_CONSTRAINTS
USER_CONS_COLUMNS	USER_INDEXES
USER_IND_COLUMNS	USER_TAB_COLUMNS
USER_OBJECTS	USER_TABLES
USER_TAB_COMMENTS	USER_VIEWS
USER_USERS	

No other Oracle database data dictionary tables or views are supported. If you use a view not on the list, you receive the Oracle database error code for no more rows available.

Queries through the gateway of any data dictionary table or view beginning with ALL_ can returns rows from the Teradata database even when access privileges for

those Teradata objects have not been granted. When querying an Oracle database with the Oracle data dictionary, rows are returned only for those objects you are permitted to access.

Data Dictionary Mapping

The tables in this section list Oracle data dictionary view names and the equivalent Teradata system tables used. A plus sign (+) indicates that a join operation is involved.

Table C–1 Oracle Data Dictionary View Names and Teradata Equivalents

View Name	Teradata System Table Name
ALL_CATALOG	DBC.TablesX + DBC.IndicesX
ALL_COL_COMMENTS	DBC.ColumnsX
ALL_CONS_COLUMNS	DBC.Indices
ALL_CONSTRAINTS	DBC.IndicesX
ALL_IND_COLUMNS	DBC.Indices + DBC.Tables + DBC.Columns
ALL_INDEXES	DBC.Indices
ALL_OBJECTS	DBC.IndicesX + DBC.TablesX
ALL_TAB_COLUMNS	DBC.ColumnsX + DBC.TablesX
ALL_TAB_COMMENTS	DBC.Tables
ALL_TABLES	DBC.TablesX
ALL_USERS	DBC.Databases
ALL_VIEWS	DBC.Tables
DBA_CATALOG	DBC.Tables + DBC.Indices
DBA_COL_COMMENTS	DBC.Columns + DBC.Tables
DBA_OBJECTS	DBC.Indices + DBC.Tables
DBA_TAB_COLUMNS	DBC.Columns + DBC.Tables
DBA_TAB_COMMENTS	DBC.Tables
DBA_TABLES	DBC.Tables
DICT_COLUMNS	DBC.Columns
DICTIONARY	DBC.Tables
DUAL	DBC.Tables
USER_CATALOG	DBC.Tables + DBC.Indices
USER_COL_COMMENTS	DBC.Columns
USER_CONS_COLUMNS	DBC.Indices
USER_CONSTRAINTS	DBC.IndicesX
USER_IND_COLUMNS	DBC.Indices + DBC.Tables + DBC.Columns
USER_INDEXES	DBC.Indices
USER_OBJECTS	DBC.Indices + DBC.Tables
USER_TAB_COLUMNS	DBC.ColumnsX + DBC.TablesX
USER_TAB_COMMENTS	DBC.TablesX

Table C–1 (Cont.) Oracle Data Dictionary View Names and Teradata Equivalents

View Name	Teradata System Table Name
USER_TABLES	DBC.Tables
USER_USERS	DBC.DatabasesX
USER_VIEWS	DBC.Tables

Default Column Values

There is a minor difference between the gateway data dictionary and a typical Oracle database data dictionary. The Oracle database columns that are missing in a Teradata system catalog table are filled with zeros, spaces, null values, not-applicable values (N.A.), or default values, depending on the column type.

Gateway Data Dictionary Descriptions

The gateway data dictionary tables and views provide the following information:

- Name, data type, and width of each column
- The contents of columns with fixed values

They are described here with information retrieved by an SQL*PLUS DESCRIBE command. The values in the Null? column might differ from the Oracle database data dictionary tables and views. Any hardcoded default value is shown to the right of an item, but this is not information returned by DESCRIBE.

Table C–2 ALL_CATALOG

Name	Type	Value
OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
TABLE_TYPE	VARCHAR2(11)	"TABLE", "VIEW", or "INDEX"

Table C–3 ALL_COL_COMMENTS

Name	Type	Value
OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(30)	-
COMMENTS	VARCHAR2(4000)	""

Table C–4 ALL_CONS_COLUMNS

Name	Type	Value
OWNER	VARCHAR2(30)	-
CONSTRAINT_NAME	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(4000)	-

Table C–4 (Cont.) ALL_CONS_COLUMNS

Name	Type	Value
POSITION	NUMBER	-

Table C–5 ALL_CONSTRAINTS

Name	Type	Value
OWNER	VARCHAR2(30)	-
CONSTRAINT_NAME	VARCHAR2(30)	-
CONSTRAINT_TYPE	VARCHAR2(1)	"R" or "P"
TABLE_NAME	VARCHAR2(30)	-
SEARCH_CONDITION	VARCHAR2(1)	""
R_OWNER	VARCHAR2(30)	""
R_CONSTRAINT_NAME	VARCHAR2(30)	""
DELETE_RULE	VARCHAR2(9)	""
STATUS	VARCHAR2(8)	""
DEFERRABLE	VARCHAR2(14)	""
DEFERRED	VARCHAR2(9)	""
VALIDATED	VARCHAR2(13)	""
GENERATED	VARCHAR2(14)	""
BAD	VARCHAR2(3)	""
RELY	VARCHAR2(4)	""
LAST_CHANGE	DATE	NULL
INDEX_OWNER	VARCHAR2(30)	""
INDEX_NAME	VARCHAR2(30)	""
INVALID	VARCHAR2(7)	""
VIEW_RELATED	VARCHAR2(14)	""

Table C–6 ALL_IND_COLUMNS

Name	Type	Value
INDEX_OWNER	VARCHAR2(30)	-
INDEX_NAME	VARCHAR2(30)	-
TABLE_OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(4000)	-
COLUMN_POSITION	NUMBER	-
COLUMN_LENGTH	NUMBER	-
DESCEND	VARCHAR2(4)	"DESC" or "ASC"

Table C–7 ALL_INDEXES

Name	Type	Value
OWNER	VARCHAR2(30)	-
INDEX_NAME	VARCHAR2(30)	-
INDEX_TYPE	VARCHAR2(27)	""
TABLE_OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
TABLE_TYPE	VARCHAR2(5)	"TABLE"
UNIQUENESS	VARCHAR2(9)	"UNIQUE" or "NONUNIQUE"
COMPRESSION	VARCHAR2(8)	""
PREFIX_LENGTH	NUMBER	0
TABLESPACE_NAME	VARCHAR2(30)	""
INI_TRANS	NUMBER	0
MAX_TRANS	NUMBER(10)	0
INITIAL_EXTENT	NUMBER(10)	0
NEXT_EXTENT	NUMBER(10)	0
MIN_EXTENTS	NUMBER(10)	0
MAX_EXTENTS	NUMBER(10)	0
PCT_INCREASE	NUMBER(10)	0
PCT_THRESHOLD	NUMBER(10)	0
INCLUDE_COLUMN	NUMBER(10)	0
FREELISTS	NUMBER(10)	0
FREELIST_GROUPS	NUMBER(10)	0
PCT_FREE	NUMBER(10)	0
LOGGING	VARCHAR2(3)	""
BLEVEL	NUMBER	0
LEAF_BLOCKS	NUMBER	0
DISTINCT_KEYS	NUMBER	-
AVG_LEAF_BLOCKS_PER_KEY	NUMBER	0
AVG_DATA_BLOCKS_PER_KEY	NUMBER	0
CLUSTERING_FACTOR	NUMBER	0
STATUS	CHAR(1)	""
NUM_ROWS	NUMBER	0
SAMPLE_SIZE	NUMBER	0
LAST_ANALYZED	DATE	NULL
DEGREE	VARCHAR2(40)	""
INSTANCES	VARCHAR2(3)	""
PARTITIONED	VARCHAR2(3)	""

Table C–7 (Cont.) ALL_INDEXES

Name	Type	Value
TEMPORARY	VARCHAR2(1)	""
GENERATED	VARCHAR2(1)	""
SECONDARY	VARCHAR2(1)	""
BUFFER_POOL	VARCHAR2(7)	""
USER_STATS	VARCHAR2(3)	""
DURATION	VARCHAR2(15)	""
PCT_DIRECT_ACCESS	NUMBER	0
ITYP_OWNER	VARCHAR2(30)	""
ITYP_NAME	VARCHAR2(30)	""
PARAMETERS	VARCHAR2(1000)	""
GLOBAL_STATS	VARCHAR2(3)	""
DOMIDX_STATUS	VARCHAR2(12)	""
DOMIDX_OPSTATUS	VARCHAR2(6)	""
FUNCIDX_STATUS	VARCHAR2(8)	""
JOIN_INDEX	VARCHAR2(3)	
IOT_REDUNDANT_PKEY_ELIM	VARCHAR2(3)	
DROPPED	VARCHAR2(3)	
VISIBILITY	VARCHAR2(9)	
DOMIDX_MANAGEMENT	VARCHAR2(14)	

Table C–8 ALL_OBJECTS

Name	Type	Value
OWNER	VARCHAR2(30)	-
OBJECT_NAME	VARCHAR2(30)	-
SUBOBJECT_NAME	VARCHAR2(30)	NULL
OBJECT_ID	NUMBER	0
DATA_OBJECT_ID	NUMBER	0
OBJECT_TYPE	VARCHAR2(19)	"TABLE" or "VIEW" or "INDEX" or "PROCEDURE"
CREATED	DATE	NULL
LAST_DDL_TIME	DATE	NULL
TIMESTAMP	VARCHAR2(19)	NULL
STATUS	VARCHAR2(7)	NULL
TEMPORARY	VARCHAR2(1)	NULL
GENERATED	VARCHAR2(1)	NULL
SECONDARY	VARCHAR2(1)	NULL

Table C–8 (Cont.) ALL_OBJECTS

Name	Type	Value
NAMESPACE	NUMBER	NULL
EDITION_NAME	VARCHAR2(30)	NULL

Table C–9 ALL_TAB_COLUMNS

Name	Type	Value
OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(30)	-
DATA_TYPE	VARCHAR2(106)	-
DATA_TYPE_MOD	VARCHAR2(3)	""
DATA_TYPE_OWNER	VARCHAR2(30)	""
DATA_LENGTH	NUMBER	-
DATA_PRECISION	NUMBER	-
DATA_SCALE	NUMBER	-
NULLABLE	VARCHAR2(1)	"Y" or "N"
COLUMN_ID	NUMBER	-
DEFAULT_LENGTH	NUMBER	0
DATA_DEFAULT	VARCHAR2(1024)	""
NUM_DISTINCT	NUMBER	0
LOW_VALUE	RAW(32)	0
HIGH_VALUE	RAW(32)	0
DENSITY	NUMBER	0
NUM_NULLS	NUMBER	0
NUM_BUCKETS	NUMBER	0
LAST_ANALYZED	DATE	NULL
SAMPLE_SIZE	NUMBER	0
CHARACTER_SET_NAME	VARCHAR2(44)	""
CHAR_COL_DECL_LENGTH	NUMBER	0
GLOBAL_STATS	VARCHAR2(3)	""
USER_STATS	VARCHAR2(3)	""
AVG_COL_LEN	NUMBER	0
CHAR_LENGTH	NUMBER	
CHAR_USED	VARCHAR2(1)	
V80_FMT_IMAGE	VARCHAR2(3)	
DATA_UPGRADED	VARCHAR2(3)	
HISTOGRAM	VARCHAR2(15)	

Table C–10 ALL_TAB_COMMENTS

Name	Type	Value
OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
TABLE_TYPE	VARCHAR2(11)	"TABLE" or "VIEW"
COMMENTS	VARCHAR2(4000)	""

Table C–11 ALL_TABLES

Name	Type	Value
OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
TABLESPACE_NAME	VARCHAR2(30)	""
CLUSTER_NAME	VARCHAR2(30)	""
IOT_NAME	VARCHAR2(8)	""
PCT_FREE	NUMBER	0
PCT_USED	NUMBER	0
INI_TRANS	NUMBER	0
MAX_TRANS	NUMBER	0
INITIAL_EXTENT	NUMBER	0
NEXT_EXTENT	NUMBER	0
MIN_EXTENTS	NUMBER	0
MAX_EXTENTS	NUMBER	0
PCT_INCREASE	NUMBER	0
FREELISTS	NUMBER	0
FREELIST_GROUPS	NUMBER	0
LOGGING	VARCHAR2(3)	""
BACKED_UP	VARCHAR2(1)	""
NUM_ROWS	NUMBER	-
BLOCKS	NUMBER	-
EMPTY_BLOCKS	NUMBER	0
AVG_SPACE	NUMBER	0
CHAIN_CNT	NUMBER	0
AVG_ROW_LEN	NUMBER	0
AVG_SPACE_FREELIST_BLOCKS	NUMBER	0
NUM_FREELIST_BLOCKS	NUMBER	0
DEGREE	VARCHAR2(10)	""
INSTANCES	VARCHAR2(10)	""
CACHE	VARCHAR2(5)	""

Table C–11 (Cont.) ALL_TABLES

Name	Type	Value
TABLE_LOCK	VARCHAR2(8)	""
SAMPLE_SIZE	NUMBER	0
LAST_ANALYZED	DATE	NULL
PARTITIONED	VARCHAR2(3)	""
IOT_TYPE	VARCHAR2(12)	""
TEMPORARY	VARCHR2(1)	""
SECONDARY	VARCHR2(1)	""
NESTED	VARCHAR2(3)	""
BUFFER_POOL	VARCHAR2(7)	""
ROW_MOVEMENT	VARCHAR2(8)	""
GLOBAL_STATS	VARCHAR2(3)	""
USER_STATS	VARCHAR2(3)	""
DURATION	VARCHAR2(15)	""
SKIP_CORRUPT	VARCHAR2(8)	""
MONITORING	VARCHAR2(3)	""
CLUSTER_OWNER	VARCHAR2(30)	
DEPENDENCIES	VARCHAR2(8)	
COMPRESSION	VARCHAR2(8)	
DROPPED	VARCHAR2(3)	

Table C–12 ALL_USERS

Name	Type	Value
USERNAME	VARCHAR2(30)	-
USER_ID	NUMBER	0
CREATED	DATE	NULL

Table C–13 ALL_VIEWS

Name	Type	Value
OWNER	VARCHAR2(30)	-
VIEW_NAME	VARCHAR2(30)	-
TEXT_LENGTH	NUMBER	0
TEXT	VARCHAR2(1)	""
TYPE_TEXT_LENGTH	NUMBER	0
TYPE_TEXT	VARCHAR2(4000)	""
OID_TEXT_LENGTH	NUMBER	0
OID_TEXT	VARCHAR2(4000)	""
VIEW_TYPE_OWNER	VARCHAR2(30)	""

Table C–13 (Cont.) ALL_VIEWS

Name	Type	Value
VIEW_TYPE	VARCHAR2(30)	""
SUPERVIEW_NAME	VARCHAR2(30)	
EDITIONING_VIEW	VARCHAR2(1)	
READ_ONLY	VARCHAR2(1)	

Table C–14 DBA_CATALOG

Name	Type	Value
OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
TABLE_TYPE	VARCHAR2(11)	"TABLE" or "VIEW"

Table C–15 DBA_COL_COMMENTS

Name	Type	Value
OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(30)	-
COMMENTS	VARCHAR2(4000)	""

Table C–16 DBA_OBJECTS

Name	Type	Value
OWNER	VARCHAR2(30)	-
OBJECT_NAME	VARCHAR2(128)	-
SUBOBJECT_NAME	VARCHAR2(30)	NULL
OBJECT_ID	NUMBER	0
DATA_OBJECT_ID	NUMBER	0
OBJECT_TYPE	VARCHAR2(19)	"TABLE" or "VIEW" or "INDEX" or "PROCEDURE"
CREATED	DATE	NULL
LAST_DDL_TIME	DATE	NULL
TIMESTAMP	VARCHAR2(19)	NULL
STATUS	VARCHAR2(7)	NULL
TEMPORARY	VARCHAR2(1)	NULL
GENERATED	VARCHAR2(1)	NULL
SECONDARY	VARCHAR2(1)	NULL
NAMESPACE	NUMBER	
EDITION_NAME	VARCHAR2(30)	

Table C–17 DBA_TAB_COLUMNS

Name	Type	Value
OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(30)	-
DATA_TYPE	VARCHAR2(106)	-
DATA_TYPE_MOD	VARCHAR2(3)	""
DATA_TYPE_OWNER	VARCHAR2(30)	""
DATA_LENGTH	NUMBER	-
DATA_PRECISION	NUMBER	-
DATA_SCALE	NUMBER	-
NULLABLE	VARCHAR2(1)	"Y" or "N"
COLUMN_ID	NUMBER	-
DEFAULT_LENGTH	NUMBER	0
DATA_DEFAULT	VARCHAR2(1024)	""
NUM_DISTINCT	NUMBER	0
LOW_VALUE	RAW(32)	0
HIGH_VALUE	RAW(32)	0
DENSITY	NUMBER	0
NUM_NULLS	NUMBER	0
NUM_BUCKETS	NUMBER	0
LAST_ANALYZED	DATE	NULL
SAMPLE_SIZE	NUMBER	0
CHARACTER_SET_NAME	VARCHAR2(44)	""
CHAR_COL_DECL_LENGTH	NUMBER	0
GLOBAL_STATS	VARCHAR2(3)	""
USER_STATS	VARCHAR2(3)	""
AVG_COL_LEN	NUMBER	0
CHAR_LENGTH	NUMBER	
CHAR_USED	VARCHAR2(1)	
V80_FMT_IMAGE	VARCHAR2(3)	
DATA_UPGRADED	VARCHAR2(3)	
HISTOGRAM	VARCHAR2(15)	

Table C–18 DBA_TAB_COMMENTS

Name	Type	Value
OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-

Table C–18 (Cont.) DBA_TAB_COMMENTS

Name	Type	Value
TABLE_TYPE	VARCHAR2(11)	"TABLE" or "VIEW"
COMMENTS	VARCHAR2(4000)	""

Table C–19 DBA_TABLES

Name	Type	Value
OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
TABLESPACE_NAME	VARCHAR2(30)	""
CLUSTER_NAME	VARCHAR2(30)	""
IOT_NAME	VARCHAR2(30)	""
STATUS	VARCHAR2(8)	
PCT_FREE	NUMBER	0
PCT_USED	NUMBER	0
INI_TRANS	NUMBER	0
MAX_TRANS	NUMBER	0
INITIAL_EXTENT	NUMBER	0
NEXT_EXTENT	NUMBER	0
MIN_EXTENTS	NUMBER	0
MAX_EXTENTS	NUMBER	0
PCT_INCREASE	NUMBER	0
FREELISTS	NUMBER	0
FREELIST_GROUPS	NUMBER	0
LOGGING	VARCHAR2(3)	""
BACKED_UP	VARCHAR2(1)	""
NUM_ROWS	NUMBER	-
BLOCKS	NUMBER	-
EMPTY_BLOCKS	NUMBER	0
AVG_SPACE	NUMBER	0
CHAIN_CNT	NUMBER	0
AVG_ROW_LEN	NUMBER	0
AVG_SPACE_FREELIST_BLOCKS	NUMBER	0
NUM_FREELIST_BLOCKS	NUMBER	0
DEGREE	VARCHAR2(10)	""
INSTANCES	VARCHAR2(10)	""
CACHE	VARCHAR2(5)	""
TABLE_LOCK	VARCHAR2(8)	""

Table C–19 (Cont.) DBA_TABLES

Name	Type	Value
SAMPLE_SIZE	NUMBER	0
LAST_ANALYZED	DATE	NULL
PARTITIONED	VARCHAR2(3)	""
IOT_TYPE	VARCHAR2(12)	""
TEMPORARY	VARCHAR2(1)	""
SECONDARY	VARCHAR2(1)	""
NESTED	VARCHAR2(3)	""
BUFFER_POOL	VARCHAR2(7)	""
ROW_MOVEMENT	VARCHAR2(8)	""
GLOBAL_STATS	VARCHAR2(3)	""
USER_STATS	VARCHAR2(3)	""
DURATION	VARCHAR2(15)	""
SKIP_CORRUPT	VARCHAR2(8)	""
MONITORING	VARCHAR2(3)	""
CLUSTER_OWNER	VARCHAR2(30)	
DEPENDENCIES	VARCHAR2(8)	
COMPRESSION	VARCHAR2(8)	
DROPPED	VARCHAR2(3)	

Table C–20 DICT_COLUMNS

Name	Type	Value
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(30)	-
COMMENTS	VARCHAR2(4000)	""

Table C–21 DICTIONARY

Name	Type	Value
TABLE_NAME	VARCHAR2(30)	-
COMMENTS	VARCHAR2(4000)	""

Table C–22 DUAL

Name	Type	Value
DUMMY	VARCHAR2(1)	"X"

Table C–23 USER_CATALOG

Name	Type	Value
TABLE_NAME	VARCHAR2(30)	-

Table C–23 (Cont.) USER_CATALOG

Name	Type	Value
TABLE_TYPE	VARCHAR2(11)	"TABLE" or "VIEW"

Table C–24 USER_COL_COMMENTS

Name	Type	Value
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(30)	-
COMMENTS	VARCHAR2(4000)	""

Table C–25 USER_CONS_COLUMNS

Name	Type	Value
OWNER	VARCHAR2(30)	-
CONSTRAINT_NAME	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(4000)	-
POSITION	NUMBER	-

Table C–26 USER_CONSTRAINTS

Name	Type	Value
OWNER	VARCHAR2(30)	-
CONSTRAINT_NAME	VARCHAR2(30)	-
CONSTRAINT_TYPE	VARCHAR2(1)	"R" or "P"
TABLE_NAME	VARCHAR2(30)	-
SEARCH_CONDITION	CHAR(1)	""
R_OWNER	VARCHAR2(30)	-
R_CONSTRAINT_NAME	VARCHAR2(30)	-
DELETE_RULE	VARCHAR2(9)	""
STATUS	VARCHAR2(8)	""
DEFERRABLE	VARCHAR2(14)	""
DEFERRED	VARCHAR2(9)	""
VALIDATED	VARCHAR2(13)	""
GENERATED	VARCHAR2(14)	""
BAD	VARCHAR2(3)	""
RELY	VARCHAR2(4)	""
LAST_CHANGE	DATE	NULL
INDEX_OWNER	VARCHAR2(30)	
INDEX_NAME	VARCHAR2(30)	
INVALID	VARCHAR2(7)	

Table C–26 (Cont.) USER_CONSTRAINTS

Name	Type	Value
VIEW_RELATED	VARCHAR2(14)	

Table C–27 USER_IND_COLUMNS

Name	Type	Value
INDEX_NAME	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(4000)	-
COLUMN_POSITION	NUMBER	-
COLUMN_LENGTH	NUMBER	-
CHAR_LENGTH	NUMBER	
DESCEND	VARCHAR2(4)	"DESC" or "ASC"

Table C–28 USER_INDEXES

Name	Type	Value
INDEX_NAME	VARCHAR2(30)	-
INDEX_TYPE	VARCHAR2(27)	""
TABLE_OWNER	VARCHAR2(30)	-
TABLE_NAME	VARCHAR2(30)	-
TABLE_TYPE	VARCHAR2(11)	"TABLE"
UNIQUENESS	VARCHAR2(9)	"UNIQUE" or "NONUNIQUE"
COMPRESSION	VARCHAR2(8)	""
PREFIX_LENGTH	NUMBER	0
TABLESPACE_NAME	VARCHAR2(30)	""
INI_TRANS	NUMBER	0
MAX_TRANS	NUMBER	0
INITIAL_EXTENT	NUMBER	0
NEXT_EXTENT	NUMBER	0
MIN_EXTENTS	NUMBER	0
MAX_EXTENTS	NUMBER	0
PCT_INCREASE	NUMBER	0
PCT_THRESHOLD	NUMBER	0
INCLUDE_COLUMN	NUMBER	0
FREELISTS	NUMBER	0
FREELIST_GROUPS	NUMBER	0
PCT_FREE	NUMBER	0
LOGGING	VARCHAR2(3)	""

Table C–28 (Cont.) USER_INDEXES

Name	Type	Value
BLEVEL	NUMBER	0
LEAF_BLOCKS	NUMBER	0
DISTINCT_KEYS	NUMBER	-
AVG_LEAF_BLOCKS_PER_KEY	NUMBER	0
AVG_DATA_BLOCKS_PER_KEY	NUMBER	0
CLUSTERING_FACTOR	NUMBER	0
STATUS	VARCHAR2(8)	""
NUM_ROWS	NUMBER	0
SAMPLE_SIZE	NUMBER	0
LAST_ANALYZED	DATE	NULL
DEGREE	VARCHAR2(40)	""
INSTANCES	VARCHAR2(40)	""
PARTITIONED	VARCHAR2(3)	""
TEMPORARY	VARCHAR2(1)	""
GENERATED	VARCHAR2(1)	""
SECONDARY	VARCHAR2(1)	""
BUFFER_POOL	VARCHAR2(7)	""
USER_STATS	VARCHAR2(3)	""
DURATION	VARCHAR2(15)	""
PCT_DIRECT_ACCESS	NUMBER	0
ITYP_OWNER	VARCHAR2(30)	""
ITYP_NAME	VARCHAR2(30)	""
PARAMETERS	VARCHAR2(1000)	""
GLOBAL_STATS	VARCHAR2(3)	""
DOMIDX_STATUS	VARCHAR2(12)	""
DOMIDX_OPSTATUS	VARCHAR2(6)	""
FUNCIDX_STATUS	VARCHAR2(8)	""
JOIN_INDEX	VARCHAR2(3)	
IOT_REDUNDANT_PKEY_ELIM	VARCHAR2(3)	
DROPPED	VARCHAR2(3)	
VISIBILITY	VARCHAR2(10)	
DOMIDX_MANAGEMENT	VARCHAR2(14)	

Table C–29 USER_OBJECTS

Name	Type	Value
OBJECT_NAME	VARCHAR2(128)	-
SUBOBJECT_NAME	VARCHAR2(30)	NULL

Table C–29 (Cont.) USER_OBJECTS

Name	Type	Value
OBJECT_ID	NUMBER	0
DATA_OBJECT_ID	NUMBER	0
OBJECT_TYPE	VARCHAR2(19)	"TABLE" or "VIEW" or "INDEX" or "PROCEDURE"
CREATED	DATE	NULL
LAST_DDL_TIME	DATE	NULL
TIMESTAMP	VARCHAR2(19)	NULL
STATUS	VARCHAR2(7)	NULL
TEMPORARY	VARCHAR2(1)	NULL
GENERATED	VARCHAR2(1)	NULL
SECONDARY	VARCHAR2(1)	NULL
NAMESPACE	NUMBER	
EDITION_NAME	VARCHAR2(30)	

Table C–30 USER_TAB_COLUMNS

Name	Type	Value
TABLE_NAME	VARCHAR2(30)	-
COLUMN_NAME	VARCHAR2(30)	-
DATA_TYPE	VARCHAR2(106)	-
DATA_TYPE_MOD	VARCHAR2(3)	""
DATA_TYPE_OWNER	VARCHAR2(30)	""
DATA_LENGTH	NUMBER	-
DATA_PRECISION	NUMBER	-
DATA_SCALE	NUMBER	-
NULLABLE	VARCHAR2(1)	"Y" or "N"
COLUMN_ID	NUMBER	-
DEFAULT_LENGTH	NUMBER	0
DATA_DEFAULT	CHAR(1)	""
NUM_DISTINCT	NUMBER	0
LOW_VALUE	RAW(32)	0
HIGH_VALUE	RAW(32)	0
DENSITY	NUMBER	0
NUM_NULLS	NUMBER	0
NUM_BUCKETS	NUMBER	0
LAST_ANALYZED	DATE	NULL
SAMPLE_SIZE	NUMBER	0

Table C–30 (Cont.) USER_TAB_COLUMNS

Name	Type	Value
CHARACTER_SET_NAME	VARCHAR2(44)	""
CHAR_COL_DECL_LENGTH	NUMBER	0
GLOBAL_STATS	VARCHAR2(3)	""
USER_STATS	VARCHAR2(3)	""
AVG_COL_LEN	NUMBER	0
CHAR_LENGTH	NUMBER	
CHAR_USED	VARCHAR2(1)	
V80_FMT_IMAGE	VARCHAR2(3)	
DATA_UPGRADED	VARCHAR2(3)	
HISTOGRAM	VARCHAR2(15)	

Table C–31 USER_TAB_COMMENTS

Name	Type	Value
TABLE_NAME	VARCHAR2(30)	-
TABLE_TYPE	VARCHAR2(11)	"TABLE" or "VIEW"
COMMENTS	VARCHAR2(4000)	""

Table C–32 USER_TABLES

Name	Type	Value
TABLE_NAME	VARCHAR2(30)	-
TABLESPACE_NAME	VARCHAR2(30)	""
CLUSTER_NAME	VARCHAR2(30)	""
IOT_NAME	VARCHAR2(30)	""
STATUS	VARCHAR2(8)	
PCT_FREE	NUMBER	0
PCT_USED	NUMBER	0
INI_TRANS	NUMBER	0
MAX_TRANS	NUMBER	0
INITIAL_EXTENT	NUMBER	0
NEXT_EXTENT	NUMBER	0
MIN_EXTENTS	NUMBER	0
MAX_EXTENTS	NUMBER	0
PCT_INCREASE	NUMBER	0
FREELISTS	NUMBER	0
FREELIST_GROUPS	NUMBER	0
LOGGING	VARCHAR2(3)	""

Table C–32 (Cont.) USER_TABLES

Name	Type	Value
BACKED_UP	VARCHAR2(1)	""
NUM_ROWS	NUMBER	-
BLOCKS	NUMBER	-
EMPTY_BLOCKS	NUMBER	0
AVG_SPACE	NUMBER	0
CHAIN_CNT	NUMBER	0
AVG_ROW_LEN	NUMBER	0
AVG_SPACE_FREELIST_BLOCKS	NUMBER	0
NUM_FREELIST_BLOCKS	NUMBER	0
DEGREE	VARCHAR2(10)	""
INSTANCES	VARCHAR2(10)	""
CACHE	VARCHAR2(5)	""
TABLE_LOCK	VARCHAR2(8)	""
SAMPLE_SIZE	NUMBER	0
LAST_ANALYZED	DATE	NULL
PARTITIONED	VARCHAR2(3)	""
IOT_TYPE	VARCHAR2(12)	""
TEMPORARY	VARCHAR2(1)	""
SECONDARY	VARCHAR2(1)	""
NESTED	VARCHAR2(3)	""
BUFFER_POOL	VARCHAR2(7)	""
ROW_MOVEMENT	VARCHAR2(8)	""
GLOBAL_STATS	VARCHAR2(3)	""
USER_STATS	VARCHAR2(3)	""
DURATION	VARCHAR2(15)	""
SKIP_CORRUPT	VARCHAR2(8)	""
MONITORING	VARCHAR2(3)	""

Table C–33 USER_USERS

Name	Type	Value
USERNAME	VARCHAR2(30)	-
USER_ID	NUMBER	0
ACCOUNT_STATUS	VARCHAR2(32)	"OPEN"
LOCK_DATE	DATE	NULL
EXPIRY_DATE	DATE	NULL
DEFAULT_TABLESPACE	VARCHAR2(30)	NULL
TEMPORARY_TABLESPACE	VARCHAR2(30)	NULL

Table C–33 (Cont.) USER_USERS

Name	Type	Value
CREATED	DATE	NULL
INITIAL_RSRC_CONSUMER_GROUP	VARCHAR2(30)	NULL
EXTERNAL_NAME	VARCHAR2(4000)	NULL

Table C–34 USER_VIEWS

Name	Type	Value
VIEW_NAME	VARCHAR2(30)	-
TEXT_LENGTH	NUMBER	0
TEXT	CHAR(1)	""
TYPE_TEXT_LENGTH	NUMBER	0
TYPE_TEXT	VARCHAR2(4000)	""
OID_TEXT_LENGTH	NUMBER	0
OID_TEXT	VARCHAR2(4000)	""
VIEW_TYPE_OWNER	VARCHAR2(30)	""
VIEW_TYPE	VARCHAR2(30)	""
SUPERVIEW_NAME	VARCHAR2(30)	
EDITIONING_VIEW	VARCHAR2(1)	

D

Initialization Parameters

The Oracle database initialization parameters in the `init.ora` file are distinct from gateway initialization parameters. Set the gateway parameters in the initialization parameter file using an agent-specific mechanism, or set them in the Oracle data dictionary using the `DBMS_HS` package. The gateway initialization parameter file must be available when the gateway is started.

This appendix contains a list of the gateway initialization parameters that can be set for each gateway and their description. It also describes the initialization parameter file syntax. It includes the following sections:

Initialization Parameter File Syntax

The syntax for the initialization parameter file is as follows:

1. The file is a sequence of commands.
2. Each command should start on a separate line.
3. End of line is considered a command terminator (unless escaped with a backslash).
4. If there is a syntax error in an initialization parameter file, none of the settings take effect.
5. Set the parameter values as follows:

   `[SET][PRIVATE]` *`parameter=value`*

   Where:

   `parameter` is an initialization parameter name. It is a string of characters starting with a letter and consisting of letters, digits and underscores. Initialization parameter names are case sensitive.

   `value` is the initialization parameter value. It is case sensitive. An initialization parameter value is either:

   a. A string of characters that does not contain any backslashes, white space or double quotation marks (")

   b. A quoted string beginning with a double quotation mark and ending with a double quotation mark. The following can be used inside a quoted string:

* backslash (\) is the escape character
* \n inserts a new line
* \t inserts a tab
* \" inserts a double quotation mark
* \\ inserts a backslash

A backslash at the end of the line continues the string on the next line. If a backslash precedes any other character then the backslash is ignored.

For example, to enable tracing for an agent, set the HS_FDS_TRACE_LEVEL initialization parameter as follows:

```
HS_FDS_TRACE_LEVEL=ON
```

SET and PRIVATE are optional keywords. You cannot use either as an initialization parameter name. Most parameters are needed only as initialization parameters, so you usually do not need to use the SET or PRIVATE keywords. If you do not specify either SET or PRIVATE, the parameter is used only as an initialization parameter for the agent.

SET specifies that, in addition to being used as an initialization parameter, the parameter value is set as an environment variable for the agent process. Use SET for parameter values that the drivers or non-Oracle system need as environment variables.

PRIVATE specifies that the initialization parameter should be private to the agent and should not be uploaded to the Oracle database. Most initialization parameters should not be private. If, however, you are storing sensitive information like a password in the initialization parameter file, then you may not want it uploaded to the server because the initialization parameters and values are not encrypted when uploaded. Making the initialization parameters private prevents the upload from happening and they do not appear in dynamic performance views. Use PRIVATE for the initialization parameters only if the parameter value includes sensitive information such as a user name or password.

SET PRIVATE specifies that the parameter value is set as an environment variable for the agent process and is also private (not transferred to the Oracle database, not appearing in dynamic performance views or graphical user interfaces).

Oracle Database Gateway for Teradata Initialization Parameters

This section lists all the initialization file parameters that can be set for the Oracle Database Gateway for Teradata. They are as follows:

- HS_DB_DOMAIN
- HS_DB_INTERNAL_NAME
- HS_DB_NAME
- HS_DESCRIBE_CACHE_HWM
- HS_LANGUAGE
- HS_LONG_PIECE_TRANSFER_SIZE
- HS_OPEN_CURSORS
- HS_RPC_FETCH_REBLOCKING
- HS_RPC_FETCH_SIZE

Initialization Parameter Description

The following sections describe all the initialization file parameters that can be set for gateways.

HS_DB_DOMAIN

Property	Description
Default value	`WORLD`
Range of values	`1` to `199` characters

Specifies a unique network sub-address for a non-Oracle system. The `HS_DB_DOMAIN` initialization parameter is similar to the `DB_DOMAIN` initialization parameter, described in the *Oracle Database Reference*. The `HS_DB_DOMAIN` initialization parameter is required if you use the Oracle Names server. The `HS_DB_NAME` and `HS_DB_DOMAIN` initialization parameters define the global name of the non-Oracle system.

Note: The `HS_DB_NAME` and `HS_DB_DOMAIN` initialization parameters must combine to form a unique address in a cooperative server environment.

HS_DB_INTERNAL_NAME

Property	Description
Default value	01010101
Range of values	1 to 16 hexadecimal characters

Specifies a unique hexadecimal number identifying the instance to which the Heterogeneous Services agent is connected. This parameter's value is used as part of a transaction ID when global name services are activated. Specifying a nonunique number can cause problems when two-phase commit recovery actions are necessary for a transaction.

HS_DB_NAME

Property	Description
Default value	HO
Range of values	1 to 8 characters

Specifies a unique alphanumeric name for the data store given to the non-Oracle system. This name identifies the non-Oracle system within the cooperative server environment. The HS_DB_NAME and HS_DB_DOMAIN initialization parameters define the global name of the non-Oracle system.

HS_DESCRIBE_CACHE_HWM

Property	Description
Default value	100
Range of values	1 to 4000

Specifies the maximum number of entries in the describe cache used by Heterogeneous Services. This limit is known as the describe cache high water mark. The cache contains descriptions of the mapped tables that Heterogeneous Services reuses so that it does not have to re-access the non-Oracle data store.

If you are accessing many mapped tables, increase the high water mark to improve performance. Increasing the high water mark improves performance at the cost of memory usage.

HS_LANGUAGE

Property	Description
Default value	System-specific
Range of values	Any valid language name (up to 255 characters)

Provides Heterogeneous Services with character set, language, and territory information of the non-Oracle data source. The value must use the following format:

language[*_territory.character_set*]

Note: The globalization support initialization parameters affect error messages, the data for the SQL Service, and parameters in distributed external procedures.

Character Sets

Ideally, the character sets of the Oracle database and the non-Oracle data source are the same. In almost all cases, HS_LANGUAGE should be set exactly the same as Oracle database character set for optimal character set mapping and performance. If they are not the same, Heterogeneous Services attempts to translate the character set of the non-Oracle data source to the Oracle database character set, and back again. The translation can degrade performance. In some cases, Heterogeneous Services cannot translate a character from one character set to another.

Note: The specified character set must be a superset of the operating system character set on the platform where the agent is installed.

As more Oracle databases and non-Oracle databases use Unicode as database character sets, it is preferable to also run the gateway in Unicode character set. To do so, you must set HS_LANGUAGE=AL32UTF8. However, when the gateway runs on Windows, the Microsoft ODBC Driver Manager interface can exchange data only in the double-byte character set, UCS2. This results in extra ratio expansion of described buffer and column sizes. Refer to HS_FDS_REMOTE_DB_CHARSET for instruction on how to adjust to correct sizes.

Language

The language component of the HS_LANGUAGE initialization parameter determines:

- Day and month names of dates
- AD, BC, PM, and AM symbols for date and time
- Default sorting mechanism

Note that Oracle does not determine the language for error messages for the generic Heterogeneous Services messages (ORA-25000 through ORA-28000). These are controlled by the session settings in the Oracle database.

Territory

The territory clause specifies the conventions for day and week numbering, default date format, decimal character and group separator, and ISO and local currency symbols. Note that the level of globalization support between the Oracle database and the non-Oracle data source depends on how the gateway is implemented.

HS_LONG_PIECE_TRANSFER_SIZE

Property	Description
Default value	64 KB
Range of values	Any value up to 2 GB

Sets the size of the piece of LONG data being transferred. A smaller piece size means less memory requirement, but more round-trips to fetch all the data. A larger piece size means fewer round-trips, but more of a memory requirement to store the intermediate pieces internally. Thus, the initialization parameter can be used to tune a system for the best performance, with the best trade-off between round-trips and memory requirements, and network latency or response time.

HS_OPEN_CURSORS

Property	Description
Default value	50
Range of values	1 to the value of Oracle's OPEN_CURSORS initialization parameter

Defines the maximum number of cursors that can be open on one connection to a non-Oracle system instance.

The value never exceeds the number of open cursors in the Oracle database. Therefore, setting the same value as the OPEN_CURSORS initialization parameter in the Oracle database is recommended.

HS_RPC_FETCH_REBLOCKING

Property	Description
Default value	ON
Range of values	OFF or ON

Controls whether Heterogeneous Services attempts to optimize performance of data transfer between the Oracle database and the Heterogeneous Services agent connected to the non-Oracle data store.

The following values are possible:

- OFF disables reblocking of fetched data so that data is immediately sent from agent to server.
- ON enables reblocking, which means that data fetched from the non-Oracle system is buffered in the agent and is not sent to the Oracle database until the amount of fetched data is equal or higher than the value of HS_RPC_FETCH_SIZE initialization parameter. However, any buffered data is returned immediately when a fetch indicates that no more data exists or when the non-Oracle system reports an error.

HS_RPC_FETCH_SIZE

Property	Description
Default value	50000
Range of values	1 to 10000000

Tunes internal data buffering to optimize the data transfer rate between the server and the agent process.

Increasing the value can reduce the number of network round-trips needed to transfer a given amount of data, but also tends to increase data bandwidth and to reduce latency as measured between issuing a query and completion of all fetches for the query. Nevertheless, increasing the fetch size can increase latency for the initial fetch results of a query, because the first fetch results are not transmitted until additional data is available.

HS_TIME_ZONE

Property	Description
Default value for '[+|-]hh:mm'	Derived from the NLS_TERRITORY initialization parameter
Range of values for '[+|-]hh:mm'	Any valid datetime format mask

Specifies the default local time zone displacement for the current SQL session. The format mask, [+|-]hh:mm, is specified to indicate the hours and minutes before or after UTC (Coordinated Universal Time—formerly Greenwich Mean Time). For example:

```
HS_TIME_ZONE = [+ | -] hh:mm
```

HS_TRANSACTION_MODEL

Property	Description
Default Value	COMMIT_CONFIRM
Range of Values	COMMIT_CONFIRM, READ_ONLY, SINGLE_SITE, READ_ONLY_AUTOCOMMIT, SINGLE_SITE_AUTOCOMMIT

Specifies the type of transaction model that is used when the non-Oracle database is updated by a transaction.

The following values are possible:

- COMMIT_CONFIRM provides read and write access to the non-Oracle database and allows the gateway to be part of a distributed update. To use the commit-confirm model, the following items must be created in the non-Oracle database:
  - Transaction log table. The default table name is HS_TRANSACTION_LOG. A different name can be set using the HS_FDS_TRANSACTION_LOG parameter. The transaction log table must be granted SELECT, DELETE, and INSERT privileges set to public.
  - Recovery account. The account name is assigned with the HS_FDS_RECOVERY_ACCOUNT parameter.
  - Recovery account password. The password is assigned with the HS_FDS_RECOVERY_PWD parameter.
- READ_ONLY provides read access to the non-Oracle database.
- SINGLE_SITE provides read and write access to the non-Oracle database. However, the gateway cannot participate in distributed updates.
- READ_ONLY_AUTOCOMMIT provides read only access to the non-Oracle database that does not use logging.

- SINGLE_SITE_AUTOCOMMIT provides read and write access to the non-Oracle database without logging. The gateway cannot participate in distributed updates. Moreover, any update to the non-Oracle database is committed immediately.

IFILE

Property	Description
Default value	None
Range of values	Valid parameter file names

Use the IFILE initialization parameter to embed another initialization file within the current initialization file. The value should be an absolute path and should not contain environment variables. The three levels of nesting limit do not apply.

See Also: *Oracle Database Reference*

HS_FDS_CONNECT_INFO

Property	Description
Default Value	None
Range of Values	Not applicable

HS_FDS_CONNECT_INFO that describes the connection to the non-Oracle system.

The default initialization parameter file already has an entry for this parameter. The syntax for HS_FDS_CONNECT_INFO for the gateway is as follows:

```
HS_FDS_CONNECT_INFO=host_alias:port_number[/database_name]
```

where, *host_alias* is the host alias name or IP address of the machine hosting the Teradata database, *port_number* is the port number of the Teradata database server, and *database_name* is the Teradata database name. The *database_name* variable is optional.

This release supports IPv6 format, so you can enter IPv6 format in place of hostname, but you need to wrap square brackets around the IPv6 specification.

For example,

```
HS_FDS_CONNECT_INFO=[2001:0db8:20c:f1ff:fec6:38af]:port_number/…
```

HS_FDS_DEFAULT_OWNER

Property	Description
Default Value	None
Range of Values	Not applicable

The name of the table owner that is used for the non-Oracle database tables if an owner is not specified in the SQL statements.

Note: If this parameter is not specified and the owner is not explicitly specified in the SQL statement, then the user name of the Oracle user or the user name specified when creating the database link is used.

HS_FDS_PROC_IS_FUNC

Property	Description
Default Value	FALSE
Range of Values	TRUE, FALSE

Enables return values from functions. By default, all stored procedures and functions do not return a return value to the user.

Note: If you set this initialization parameter, you must change the syntax of the procedure execute statement for all existing stored procedures to handle return values.

HS_FDS_RECOVERY_ACCOUNT

Property	Description
Default Value	RECOVER
Range of values	Any valid user ID

Specifies the name of the recovery account used for the commit-confirm transaction model. An account with user name and password must be set up at the non-Oracle system. For more information about the commit-confirm model, see the HS_TRANSACTION_MODEL parameter.

The name of the recovery account is case sensitive.

HS_FDS_RECOVERY_PWD

Property	Description
Default Value	RECOVER
Range of values	Any valid password

Specifies the password of the recovery account used for the commit-confirm transaction model set up at the non-Oracle system. For more information about the commit-confirm model, see the HS_TRANSACTION_MODEL parameter.

The name of the password of the recovery account is case sensitive.

HS_FDS_TRACE_LEVEL

Property	Description
Default Value	OFF
Range of values	OFF, ON, DEBUG

Specifies whether error tracing is turned on or off for gateway connectivity.

The following values are valid:

- OFF disables the tracing of error messages.
- ON enables the tracing of error messages that occur when you encounter problems. The results are written by default to a gateway log file in LOG directory where the gateway is installed.
- DEBUG enables the tracing of detailed error messages that can be used for debugging.

HS_FDS_TRANSACTION_LOG

Property	Description
Default Value	HS_TRANSACTION_LOG
Range of Values	Any valid table name

Specifies the name of the table created in the non-Oracle system for logging transactions. For more information about the transaction model, see the HS_TRANSACTION_MODEL parameter.

HS_FDS_FETCH_ROWS

Property	Description
Default Value	100
Range of Values	Any integer between 1 and 1000
Syntax	HS_FDS_FETCH_ROWS=*num*

HS_FDS_FETCH_ROWS specifies the fetch array size. This is the number of rows to be fetched from the non-Oracle database and to return to Oracle database at one time. This parameter will be affected by the HS_RPC_FETCH_SIZE and HS_RPC_FETCH_REBLOCKING parameters.

HS_IDLE_TIMEOUT

Property	Description
Default Value	0 (no timeout)
Range of Values	0-9999 (minutes)
Syntax	HS_IDLE_TIMEOUT=*num*

This feature is only available for Oracle Net TCP protocol. When there is no activity for a connected gateway session for this specified time period, the gateway session would be terminated automatically with pending update (if any) rolled back.

HS_NLS_LENGTH_SEMANTICS

Property	Description
Default Value	BYTE
Range of Values	BYTE | CHAR
Syntax	HS_NLS_LENGTH_SEMANTICS = { BYTE | CHAR }

This release of gateway has Character Semantics functionality equivalent to the Oracle Database Character Semantics, that is, NLS_LENGTH_SEMANTICS. When HS_NLS_LENGTH_SEMANTICS is set to CHAR, the (VAR)CHAR columns of Teradata are to be interpreted as having CHAR semantics. The only situation the gateway does not honor the HS_NLS_LENGTH_SEMANTICS=CHAR setting is when both Oracle database and the gateway are on the same multi-byte character set

HS_KEEP_REMOTE_COLUMN_SIZE

Property	Description
Default Value	OFF
Range of Values	OFF | LOCAL | REMOTE |ALL
Syntax	HS_KEEP_REMOTE_COLUMN_SIZE = OFF | LOCAL | REMOTE | ALL
Parameter type	String

HS_KEEP_REMOTE_COLUMN_SIZE specifies whether to suppress ratio expansion when computing the length of (VAR)CHAR datatypes during data conversion from non-Oracle database to the gateway, and then to the Oracle database. When it is set to REMOTE, the expansion is suppressed between the non-Oracle database and the gateway. When it is set to LOCAL, the expansion is suppressed between the gateway and the Oracle database. When it is set to ALL, the expansion is suppressed from the non-Oracle database to the Oracle database.

When the parameter is set, the expansion is suppressed when reporting the remote column size, calculating the implicit resulting buffer size, and instantiating in the local Oracle database. This has effect only for remote column size from non-Oracle database to the Oracle database. If the gateway runs on Windows and HS_LANGUAGE=AL32UTF8, then you must not specify this parameter, as it would influence other ratio related parameter operation. It has no effect for calculating ratio for data moving from Oracle database to non-Oracle database through gateway during INSERT, UPDATE, or DELETE.

HS_FDS_REMOTE_DB_CHARSET

Property	Description
Default Value	None

Property	Description
Range of values	Not applicable
Syntax	HS_FDS_REMOTE_DB_CHARSET

This parameter is valid only when HS_LANGUAGE is set to AL32UTF8 and the gateway runs on Windows. As more Oracle databases and non-Oracle databases use Unicode as database character sets, it is preferable to also run the gateway in Unicode character set. To do so, you must set HS_LANGUAGE=AL32UTF8. However, when the gateway runs on Windows, the Microsoft ODBC Driver Manager interface can exchange data only in the double-byte character set, UCS2. This results in extra ratio expansion of described buffer and column sizes. To compensate, the gateway can re-adjust the column size if HS_FDS_REMOTE_DB_CHARSET is set to the corresponding non-Oracle database character set. For example, HS_FDS_REMOTE_DB_CHARSET=KO16KSC5601.

HS_FDS_SUPPORT_STATISTICS

Property	Description
Default Value	TRUE
Range of values	{TRUE|FALSE}
Syntax	HS_FDS_SUPPORT_STATISTICS= {TRUE|FALSE}

We gather statistics from the non-Oracle database by default. You can choose to disable the gathering of remote database statistics by setting the HS_FDS_SUPPORT_STATISTICS parameter to FALSE.

HS_FDS_SQLLEN_INTERPRETATION

Property	Description
Default Value	64
Range of values	{64|32}
Syntax	HS_FDS_SQLLEN_INTERPRETATION= {64|32}

This parameter is only valid for 64 bit platforms. ODBC standard specifies SQLLEN (of internal ODBC construct) being 64 bit on 64 bit platforms, but some ODBC driver managers and drivers violate this convention, and implement it as 32 bit. In order for the gateway to compensate their behavior, you need to specify HS_FDS_SQLLEN_INTERPRETATION=32 if you use these types of driver managers and driver.

Index

I

K

L

N

O

P

R

S

T

U

V

W