Contents

Preface

Intended Audience xi
Documentation Accessibility xi
Related Documents xi
Typographic Conventions xii
SQL*Plus Prompts xii
Storage Measurements xii

1 Introduction to the Oracle Database Gateway for DRDA

1.1 Overview of the Oracle Database Gateway for DRDA 1-1
1.2 Gateway Capabilities 1-2
  1.2.1 Transparency at All Levels 1-2
  1.2.2 Extended Database Services 1-3
  1.2.3 Extended Advanced Networking, Internet and Intranet Support 1-3
  1.2.4 Dynamic Dictionary Mapping 1-4
  1.2.5 SQL 1-4
  1.2.6 Data Definition Language 1-4
  1.2.7 Data Control Language 1-5
  1.2.8 Passthrough and Native DB2 SQL 1-5
  1.2.9 Stored Procedures 1-5
  1.2.10 Languages 1-5
  1.2.11 Oracle Database Technology and Tools 1-5
  1.2.12 SQL*Plus 1-6
  1.2.13 Two-Phase Commit and Multi-Site Transactions 1-6
  1.2.14 Site Autonomy 1-6
  1.2.15 Migration and Coexistence 1-6
  1.2.16 Security 1-6
  1.2.17 DRDA UDB Server Encryption support 1-7
1.3 Terms 1-7
1.4 Architecture 1-7
1.5 Implementation 1-8
1.6 How the Gateway Works 1-9
2 Release Information

2.1 Product Set 2-1
2.2 Changes and Enhancements 2-1
   2.2.1 Remote Insert Rowsource 2-1
   2.2.2 Gateway Password Encryption Tool 2-2
   2.2.3 Result Sets and Stored Procedures 2-2
2.3 Product Migration 2-3
2.4 Known Problems 2-3
2.5 Known Restrictions 2-4
   2.5.1 DB2 Considerations 2-4
      2.5.1.1 DD Basic Tables and Views 2-4
      2.5.1.2 SUBSTR Function Post-Processed 2-4
      2.5.1.3 Data type Limitations 2-4
      2.5.1.4 Null Values and Stored Procedures 2-4
      2.5.1.5 String Concatenation of Numbers 2-4
      2.5.1.6 GLOBAL_NAMES Initialization Parameter 2-4
      2.5.1.7 DRDA Package and DB2 considerations 2-5
      2.5.1.8 Date Arithmetic 2-5
      2.5.1.9 Row Length Limitation 2-5
      2.5.1.10 LONG Data type in SQL*Plus 2-5
      2.5.1.11 Stored Procedures and Transaction Integrity 2-5
   2.5.2 SQL Limitations 2-6
      2.5.2.1 Oracle ROWID Column 2-6
      2.5.2.2 Oracle Bind Variables 2-6
      2.5.2.3 CONNECT BY Is Not Supported 2-6

3 Using the Oracle Database Gateway for DRDA

3.1 DRDA Gateway Features 3-1
   3.1.1 CHAR Semantics 3-1
   3.1.2 Multi-byte Character Sets Ratio Suppression 3-1
   3.1.3 IPv6 Support 3-2
   3.1.4 Gateway Session IDLE Timeout 3-2
3.2 Processing a Database Link 3-2
   3.2.1 Creating Database Links 3-2
   3.2.2 Dropping Database Links 3-3
   3.2.3 Examining Available Database Links 3-4
3.2.4 Limiting the Number of Active Database Links 3-4
3.3 Accessing the Gateway 3-4
3.4 Accessing i5/OS File Members 3-5
3.5 Using the Synonym Feature 3-5
3.6 Performing Distributed Queries 3-5
3.6.1 Two-Phase Commit Processing 3-6
3.6.2 Distributed DRDA Transactions 3-7
3.7 Replicating in a Heterogeneous Environment 3-7
3.8 Copying Data from Oracle Database to DRDA Server 3-7
3.9 Copying Data from DRDA Server to Oracle Database 3-8
3.10 Tracing SQL Statements 3-8

4 Developing Applications

4.1 Gateway Appearance to Application Programs 4-1
4.1.1 Fetch Reblocking 4-2
4.2 Using Oracle Stored Procedures with the Gateway 4-2
4.3 Using DRDA Server Stored Procedures with the Gateway 4-3
4.3.1 Oracle Application and DRDA Server Stored Procedure Completion 4-4
4.3.2 Procedural Feature Considerations with DB2 4-5
4.3.3 Result Sets and Stored Procedures 4-5
4.3.3.1 OCI Program Fetching from Result Sets in Sequential Mode 4-6
4.3.3.2 PL/SQL Program Fetching from Result Sets in Sequential Mode 4-8
4.4 Database Link Behavior 4-9
4.5 Oracle Database SQL Construct Processing 4-9
4.5.1 Compatible SQL Functions 4-9
4.5.2 Translated SQL Functions 4-9
4.5.3 Compensated SQL Functions 4-9
4.5.3.1 Post-Processing 4-10
4.5.4 Native Semantic SQL Functions 4-10
4.5.5 DB2 UDB for z/OS SQL Compatibility 4-10
4.5.6 DB2 UDB for Unix, Linux, and Windows Compatibility 4-13
4.5.7 DB2 UDB for iSeries Compatibility 4-16
4.6 Native Semantics 4-18
4.6.1 SQL Functions That Can Be Enabled 4-19
4.6.2 SQL Functions That Can Be Disabled 4-21
4.6.3 SQL Set Operators and Clauses 4-21
4.7 DRDA Data type to Oracle Data type Conversion 4-21
4.7.1 Performing Character String Operations 4-22
4.7.2 Converting Character String Data types 4-23
4.7.3 Performing Graphic String Operations 4-23
B Initialization Parameters

B.1 Initialization Parameter File Syntax
B.2 Oracle Database Gateway for DRDA Initialization Parameters
B.3 HS_CALL_NAME
B.4 HS_DB_DOMAIN
B.5 HS_DB_INTERNAL_NAME
B.6 HS_DB_NAME
B.7 HS_DESCRIBE_CACHE_HWM
B.8 HS_LANGUAGE
  B.8.1 Character Sets
  B.8.2 Language
  B.8.3 Territory
B.9 HS_LONG_PIECE_TRANSFER_SIZE
B.10 HS_OPEN_CURSORS
B.11 HS_RPC_FETCH_REBLOCKING
B.12 HS_RPC_FETCH_SIZE
B.13 HS_TRANSACTION_MODEL
B.14 IFILE
Index
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>The Gateway Architecture</td>
<td>1-8</td>
</tr>
<tr>
<td>4-1</td>
<td>Calling Oracle Stored Procedures in a Distributed Oracle Environment</td>
<td>4-3</td>
</tr>
<tr>
<td>4-2</td>
<td>Running DRDA Server Stored Procedures</td>
<td>4-4</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>4-1</td>
<td>SQL Compatibility, by Oracle SQL function</td>
<td>4-10</td>
</tr>
<tr>
<td>4-2</td>
<td>DB2 UDB for Unix, Linux, and Windows Compatibility, by Oracle SQL Function</td>
<td>4-13</td>
</tr>
<tr>
<td>4-3</td>
<td>DB2 UDB for iSeries Compatibility, by Oracle SQL Function</td>
<td>4-16</td>
</tr>
<tr>
<td>4-4</td>
<td>Data Type Mapping and Restrictions</td>
<td>4-22</td>
</tr>
</tbody>
</table>
Preface

The Oracle Database Gateway for DRDA provides users with transparent access to DB2.

Intended Audience

This guide is intended for anyone responsible for installing, configuring, and administering the gateway, and also for application developers.

Read this guide if you are responsible for writing applications that access DRDA databases through the gateway.

You must understand the fundamentals of Oracle Database Gateway and the operating system you are working on 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 that have purchased support 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

Oracle Database Heterogeneous Connectivity User's Guide
Oracle Database Administrator's Guide
Oracle Database Concepts
Oracle Database Error Messages
Oracle Database Performance Tuning Guide
Oracle Database Security Guide
Typographic Conventions

The following typographic conventions are used in this guide:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands, directory names, user names, path names, and file names.</td>
</tr>
<tr>
<td>italics</td>
<td>Italic type indicates variables, including variable portions of file names. It is also used for emphasis and for book titles.</td>
</tr>
<tr>
<td>UPPERCASE</td>
<td>Uppercase letters indicate Structured Query Language (SQL) reserved words, initialization parameters, and environment variables.</td>
</tr>
<tr>
<td>Bold</td>
<td>Bold type indicates screen names and fields.</td>
</tr>
<tr>
<td>SQL*Plus prompts</td>
<td>The SQL<em>Plus prompt, SQL&gt;, appears in SQL statement and SQL</em>Plus command examples. Enter your response at the prompt. Do not enter the text of the prompt, &quot;SQL&gt;&quot; in your response.</td>
</tr>
</tbody>
</table>

SQL*Plus Prompts

The SQL*Plus prompt, SQL>, appears in SQL statements and SQL*Plus command examples. Enter your response at the prompt. Do not enter the text of the prompt, "SQL>", in your response.

Storage Measurements

Storage measurements use the following abbreviations:

- KB, for kilobyte, which equals 1,024 bytes
- MB, for megabyte, which equals 1,048,576 bytes
- GB, for gigabyte, which equals 1,073,741,824 bytes
1 Introduction to the Oracle Database Gateway for DRDA

The following sections provide information about the architecture, uses, and features of the Oracle Database Gateway for DRDA:

• Overview of the Oracle Database Gateway for DRDA
• Gateway Capabilities
• Terms
• Architecture
• Implementation
• How the Gateway Works
• Oracle Tools and the Gateway
• Features

1.1 Overview of the Oracle Database Gateway for DRDA

The Oracle Database Gateway for DRDA gives you access to your Oracle data and DB2 data with a single set of applications while you continue to use existing IBM applications to access your DB2 data. The gateway enables you to:

• Integrate heterogeneous database management systems so that they appear as a single homogeneous database system.
• Read and write data from Oracle applications to data in DB2 UDB for z/OS, DB2 Universal Database™ for iSeries™ (DB2 UDB for iSeries), and DB2 Universal Database.

Oracle Database 12c Release 2 (12.2) provides the foundation for the next generation of the Oracle Database Gateways, which delivers enhanced integration capabilities by exploiting Oracle Database Heterogeneous Services.

As an integrated component of the Oracle database, Heterogeneous Services can take advantage of the powerful SQL parsing and distributed optimization capabilities of the Oracle database. This integration also ensures that the gateway can immediately take advantage of any enhancements made to future releases of the Oracle database. For detailed information on Oracle Heterogeneous Services, refer to Oracle Database Heterogeneous Connectivity User's Guide.

The gateways are even more tightly integrated with Oracle Database 12c Release 2 (12.2) than previous versions, enabling improved performance and enhanced functionality while still providing transparent integration of Oracle and non-Oracle data. For example, connection initialization information is available in the local Oracle database, reducing the number of round trips and the amount of data sent over the network. SQL execution is also faster, because, statements issued by an application are parsed and translated once and can then be reused by multiple applications.
1.2 Gateway Capabilities

Oracle Database Gateway for DRDA enables you to integrate your heterogeneous system into a single, seamless environment. If data is moved from a DRDA database to an Oracle database, then no changes in application design or function are needed. The gateway handles all differences in both data types and SQL functions between the application and the database. As a result, end users and application programmers are not required to know either the physical location or the storage characteristics of the data.

This transparency not only enables you to integrate heterogeneous data seamlessly, it also simplifies your gateway implementation, application development, and maintenance. The gateway capabilities are as follows:

- Transparency at All Levels
- Extended Database Services
- Extended Advanced Networking Internet and Intranet Support
- Dynamic Dictionary Mapping
- SQL
- Data Definition Language
- Data Control Language
- Passthrough and Native DB2 SQL
- Stored Procedures
- Languages
- Oracle Database Technology and Tools
- SQL*Plus
- Two-Phase Commit and Multi-Site Transactions
- Site Autonomy
- Migration and Coexistence
- Security
- DRDA UDB Server Encryption support

1.2.1 Transparency at All Levels

Oracle Database Gateway for DRDA gives you transparency at the following levels:

- Location
  Users can access tables by name and do not need to know the physical location of the tables.
- Network
  The gateway exploits the Oracle Net technology to allow users to access data across multiple networks without concern for the network architecture. TCP/IP protocol is supported. This release supports IPV4 and IPV6 between Oracle database and the gateway, and also between the gateway and the DB2 server.
• Operating System

Users can access data stored under multiple operating systems without being aware of the operating systems that hold the data.

• Data Storage

The gateway provides the ability for data to be accessed regardless of the database or file format.

• Access Method

You can utilize a single dialect of SQL for any data store, eliminating the need to code for database-specific access methods or SQL implementations.

1.2.2 Extended Database Services

Following are some of the Oracle database services available through the gateway:

• SQL functions

Your application can access all your data using Oracle SQL. The method by which the gateways are integrated with the Oracle database ensures that the latest features of each database release are always available immediately to the gateway.

• Distributed capabilities

Heterogeneous data can be integrated seamlessly because Oracle distributed capabilities, such as JOIN and UNION, can be applied against non-Oracle data without any special programming or mapping.

• Distributed query optimization

The Oracle database can utilize its advanced query optimization techniques to ensure that SQL statements are executed efficiently against any of your data. The data distribution and storage characteristics of local and remote data are equally considered.

• Two-phase commit protection

The Oracle database two-phase commit mechanism provides consistency across data stores by ensuring that a transaction that spans data stores is still treated as a single unit of work. Changes are not committed (or permanently stored) in any data store unless the changes can be committed in all data stores that will be affected.

• Stored procedures and database triggers

The same Oracle stored procedures and database triggers can be used to access all of your data, thereby ensuring uniform enforcement of your business rules across the enterprise.

1.2.3 Extended Advanced Networking, Internet and Intranet Support

The gateway integration with the Oracle database extends the benefits of the Oracle Internet and Oracle Net software to non-Oracle data and extends the benefits of the Oracle client/server and server/server connectivity software. These features include:

• Application server support
Any Internet or intranet application that can access data in the Oracle database can also incorporate information from data stores accessible through the gateways. Web browsers can connect to the Oracle database using any application server product that supports Oracle software.

- Implicit protocol conversion
  Oracle and Oracle Net can work together as a protocol converter, allowing applications to transparently access other data stores on platforms that do not support the clients network protocol. An Oracle database can use TCP/IP to communicate with the gateway and another data store.

- Advanced Security
  Non-Oracle data can be protected from unauthorized access or tampering during transmission to the client. You can do this by using the hardware-independent and protocol-independent encryption and checksum services of Advanced Security.

- Wireless communication
  Oracle Mobile Agents, an Oracle industry-leading mobile technology, enables wireless communication to Oracle database or to any databases that are accessible through the gateways. This gives your field personnel direct access to enterprise data from mobile laptop computers.

### 1.2.4 Dynamic Dictionary Mapping

The simple setup of the gateway does not require any additional mapping. Before an application can access any information, the application must be told the structure of the data, such as the columns of a table and their lengths. Many products require administrators to manually define that information in a separate data dictionary stored in a hub. Applications then access the information using the hub dictionary instead of the native dictionaries of each database. This approach requires a great deal of manual configuration and maintenance on your part. As administrators, you must update the data dictionary in the hub whenever the structure of a remote table is changed.

Inefficient duplication is not necessary with Oracle Database Gateway for DRDA. The gateway uses the existing native dictionaries of each database. Your applications access data using the dictionaries designed specifically for each database, which means no redundant dictionary ever needs to be created or maintained.

### 1.2.5 SQL

Oracle Database Gateways ease your application development and maintenance by allowing you to access any data using a uniform set of SQL. Changes to the location, storage characteristics, or table structure do not require any changes to your applications. ANSI and ISO standard SQL are supported, along with powerful Oracle extensions.

### 1.2.6 Data Definition Language

Oracle Applications can create tables in target data stores by using native data definition language (DDL) statements.
1.2.7 Data Control Language

You can issue native data control language (DCL) statements from an Oracle environment, allowing central administration of user privileges and access levels for heterogeneous data stores.

1.2.8 Passthrough and Native DB2 SQL

Execution of native DB2 SQL can be passed through the gateway for execution directly against DB2. This enables applications to send statements, such as a DB2 CREATE TABLE, to the gateway for execution on a target DB2 system.

1.2.9 Stored Procedures

The gateway enables you to exploit both Oracle and non-Oracle stored procedures, leveraging your investments in a distributed, multi-database environment. Oracle stored procedures can access multiple data stores easily, without any special coding for heterogeneous data access.

**Oracle Stored Procedures**

Oracle stored procedures enable you to access and update DB2 data using centralized business rules stored in the Oracle database. Using Oracle stored procedures can increase your database performance by minimizing network traffic. Instead of sending individual SQL statements across the network, an application can send a single EXECUTE command to begin an entire PL/SQL routine.

**Native DB2 Stored Procedures**

The gateway can execute DB2 stored procedures using standard Oracle PL/SQL. The Oracle application executes the DB2 stored procedure as if it were an Oracle remote procedure.

1.2.10 Languages

Any application or tool that supports the Oracle database can access over thirty different data sources through the Oracle gateways. A wide variety of open system tools from Oracle Corporation and third-party vendors can be used, even if the data is stored in legacy, proprietary formats. Hundreds of tools are supported, including ad hoc query tools, Web browsers, turnkey applications, and application development tools.

1.2.11 Oracle Database Technology and Tools

The gateway is integrated into the Oracle database technology, which provides global query optimization, transaction coordination for multi-site transactions, and support for all Oracle Net configurations. Tools and applications that support the Oracle database can be used to access heterogeneous data through the gateway.
1.2.12 SQL*Plus

You can use SQL*Plus for moving data between databases. This product gives you the ability to copy data from your department databases to corporate Oracle database instances.

1.2.13 Two-Phase Commit and Multi-Site Transactions

The gateway can participate as a partner in multi-site transactions and two-phase commit. How this occurs depends on the capabilities of the underlying data source, meaning that the gateway can be implemented as any one of the following:

- Full two-phase commit partner
- Commit point site
- Single-site update partner
- Read-only partner

The deciding factors for the implementation of the gateway are the locking and transaction-handling capabilities of your target database.

Oracle Database Gateway for DRDA by default is configured as a commit point site, that is, commit confirm protocol. Optionally, you can configure the gateway as read-only if you choose to enforce read-only capability through the gateway. Other protocols are not supported.

1.2.14 Site Autonomy

All Oracle database products, including gateways, supply site autonomy. For example, administration of a data source remains the responsibility of the original system administrator. Site autonomy also functions such that gateway products do not override the security measures established by the data source or operating environment.

1.2.15 Migration and Coexistence

The integration of a data source through the gateway does not require any changes to be made to applications at the data source. The result is that the Oracle database technology is non-intrusive, providing coexistence and an easy migration path.

1.2.16 Security

The gateway does not bypass existing security mechanisms. Gateway security coexists with the security mechanisms already used in the operating environment of the data source.

Functionally, gateway security is identical to that of an Oracle database, as described in the Oracle Database Security Guide. Oracle database security is mapped to the data dictionary of the data source.
1.2.17 DRDA UDB Server Encryption support

This release of Gateway for DRDA provides complete UDB server Encryption support. Refer to the following new parameters for various options:

• HS_FDS_AUTHENTICATE_USER
• HS_FDS_ENCRYPT_SESSION
• HS_FDS_TRUSTSTORE_FILE
• HS_FDS_TRUSTSTORE_PASSWORD

1.3 Terms

The terms used in this guide do not necessarily conform to the IBM terminology. The following list presents several terms and their meanings as used within this guide:

DRDA data is, generically, any database data accessed through DRDA.

DRDA database is the collection of data that belongs to a DRDA server.

DRDA server is a database server that can be accessed through DRDA. IBM terminology for a DRDA server is a DRDA Application Server, or AS.

DRDA server type is a specific database product or program that can act as a DRDA server.

Oracle database is any Oracle Database 12c Release 2 (12.2) instance that communicates with the Oracle Database Gateway for DRDA to distribute database access operations to a DRDA server. The Oracle database can also be used for non-gateway applications.

DB2 Universal Database is a generic name for all implementations of IBM's DB2 Database product. DB2/UDB is frequently used as an abbreviation for the DB2 Universal Database for Unix, Linux, and Windows product.

1.4 Architecture

The Oracle Database Gateway for DRDA works with the Oracle database to shield most of the differences of the non-Oracle database from Oracle applications.

The architecture consists of the following main components:

• Client
  The client is an Oracle application or tool.

• Oracle database
  The gateway instance is accessed by an Oracle database with procedural and distributed options. Usually, the Oracle database is installed on the same host as the gateway, but this is not a requirement. The Oracle database and the gateway communicate in the normal Oracle database-to-server manner.

  If the Oracle database is not on the host where the gateway resides, then you must install the correct Oracle networking software on the platform where the server resides. For Oracle database, you must install Oracle Net on the Oracle database machine.
• Oracle Database Gateway for DRDA
The gateway must be installed on hosts that are running the appropriate operating system.
If the Oracle database is not on the same host, then you must also install Oracle Net so that the gateway and Oracle database can communicate.

• DRDA server
The DRDA server must be on a system accessible to the host via a network.

Multiple Oracle databases can access the same gateway.

Figure 1-1 illustrates the gateway architecture.

Figure 1-1 The Gateway Architecture

1.5 Implementation
When the gateway is installed on your host, it has some of the same components as an Oracle database instance on your host. The gateway has the following components:

• A base file directory, similar to the one associated with an Oracle instance
  ORACLE_HOME environment variable
• A gateway system identifier (SID), comparable to an Oracle instance ORACLE_SID
• Oracle Net to support communication between the Oracle database and the Oracle Database Gateway for DRDA

The gateway does not have:

• Control, redo log, or database files
• The full set of subdirectories and ancillary files that are associated with an installed Oracle database

Because the gateway does not have background processes and does not need a management utility, such as Oracle Enterprise Manager, you do not need to start the gateway product. Each Oracle database user session that accesses a particular
gateway creates an independent process on the host. This process runs the gateway session and executes network operations to communicate with a DRDA server.

1.6 How the Gateway Works

The gateway has no database functions of its own. Instead, it provides an interface by which an Oracle database can direct part or all of a SQL operation to a DRDA database.

The gateway that is supporting the DRDA server is identified to the Oracle database using a database link. The database link is the same construct that is used to identify other Oracle databases. Tables on the DRDA server are referenced in SQL as:

`table_name@dblink_name`

or

`owner.table_name@dblink_name`

If you create synonyms or views in the Oracle database, then you can refer to tables on the DRDA server by using simple names as though the table were local to the Oracle database.

When the Oracle database encounters a reference to a table that is on the DRDA server, the applicable portion of the SQL statement is sent to the gateway for processing. Any host variables that are associated with the SQL statement are bound to the gateway and, therefore, to the DRDA server.

The gateway is responsible for sending these SQL statements to the DRDA server for execution and for fielding and returning responses. The responses are either data or messages. Any conversions between Oracle data types and DRDA data types are performed by the gateway. Both the Oracle database and the application read and process only Oracle data types.

SQL Differences

Not all SQL implementations are the same. The Oracle database supports a larger set of built-in functions than the databases that are currently accessed through the gateway. The Oracle database and the gateway work together to convert SQL to a form that is compatible with the specific DRDA server.

During this conversion, an Oracle database function can be converted to a function that is recognizable to the specific DRDA server. For example, the Oracle database NVL function is converted to the DB2 VALUE function.

Alternatively, the Oracle database withholds functions that are not executable by the DRDA server and performs them after rows are fetched from the DRDA database. This processing only applies to SELECT statements. The Oracle database and the gateway cannot perform this kind of manipulation on UPDATE, INSERT, or DELETE statements because doing so changes transaction semantics.

1.7 Oracle Tools and the Gateway

Use the Oracle Database Gateway to run applications, such as Oracle database tools, that read and write data that is stored in DRDA databases.
While the Oracle Database Gateway for DRDA provides no new application or development facilities, it extends the reach of existing Oracle database tools to include data in non-Oracle databases that support DRDA.

Using the Oracle Database Gateway for DRDA with other Oracle products can greatly extend the capabilities of the stand-alone gateway.

**SQL*Plus**

Use SQL*Plus and the Oracle Database Gateway for DRDA to create a distributed database system, providing an easy-to-use transfer facility for moving data between the distributed databases. One possible use is to distribute the data in your corporate Oracle database to departmental DRDA databases. You can also distribute data in your corporate DRDA database to departmental Oracle databases.

### 1.8 Features

Following is a list of important features that characterize Oracle Database Gateway for DRDA:

- Heterogeneous Services Architecture
- Performance Enhancements
- Fetch Reblocking
- Oracle Database Passthrough Supported
- Retrieving Result Sets Through Passthrough
- Support for TCP/IP
- Native Semantics
- Columns Supported in a Result Set
- EXPLAIN_PLAN Improvement
- Heterogeneous Database Integration
- Minimum Impact on Existing Systems
- Application Portability
- Remote Data Access
- Support for Distributed Applications
- Application Development and End User Tools
- Password Encryption Utility
- Support for DB2 UDB for z/OS Stored Procedures
- IBM DB2 Universal Database Support
- DB2 z/OS ASCII and UNICODE Table Support
- Read-Only Support
- Support for Graphic and Multi-byte Data
- Support for DB2 Universal Database on Intel Hardware
- Data Dictionary Support for DB2 Universal Database
Heterogeneous Services Architecture

Oracle Database Gateway for DRDA utilizes the Oracle Heterogeneous Services component within the Oracle database. Heterogeneous Services is the building block for the next generation of Oracle database gateways.

For detailed information about heterogeneous services, refer to Oracle Database Heterogeneous Connectivity User's Guide.

Performance Enhancements

Oracle Database Gateway for DRDA contains several internal performance enhancements. This product has shown major improvements in response time and CPU utilization for all relevant address spaces for a variety of workloads compared to version 10 gateways. The actual performance improvement at your site might vary, depending on your installation type and workload.

Fetch Reblocking

The array size of the application for SELECT is effective between the application and the Oracle database. However, the array blocksize and the block fetch between the Oracle database and the gateway are controlled by two Heterogeneous Services initialization parameters: HS_RPC_FETCH_SIZE and HS_RPC_FETCH_REBLOCKING. These parameters are specified in the gateway initialization file. Refer to Oracle Database Heterogeneous Connectivity User's Guide for more information.

Oracle Database Passthrough Supported

You can use the Oracle database DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE method to pass commands or statements available in your DRDA server through the gateway.

Retrieving Result Sets Through Passthrough

Oracle Database Gateway for DRDA provides a facility to retrieve result sets from a SELECT statement issued with passthrough. Refer to "Retrieving Results Sets Through Passthrough" for additional information.

Support for TCP/IP

This release of the gateway only supports the TCP/IP communication protocol between the gateway and the DRDA server. Refer to Oracle Database Gateway Installation and Configuration Guide for IBM AIX on POWER Systems (64-Bit), Linux x86-64, Oracle Solaris on SPARC (64-Bit), Oracle Solaris on x86-64 (64-Bit) and HP-UX Itanium, or Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows depending on your platform.

Native Semantics

This release of the gateway supports the ability to selectively enable or disable post-processing of various SQL functions by the DRDA server. Refer to "Native Semantics" for further information.

Columns Supported in a Result Set

Oracle Database Gateway for DRDA supports up to 1000 columns in a result set.
EXPLAIN_PLAN Improvement

The EXPLAIN_PLAN table contains the actual SQL statements passed to the DRDA server from the Oracle database through the gateway.

Heterogeneous Database Integration

The gateway support for ANSI-standard SQL enables read/write access to DRDA databases. Even if your data exists on different platforms in different applications, new applications can use all data, regardless of location.

Minimum Impact on Existing Systems

The gateway does not require installation of additional Oracle software on your OS/390 (MVS), AS/400, UNIX based, or Microsoft Windows target system. The database interface that it uses is provided by IBM and is built into the DRDA database products and network facilities that already exist on these platforms.

Configuring an IBM system for DRDA access typically consists of defining the network resources involved and establishing access security definitions specific to the target database.

Application Portability

The gateway’s ability to interface with heterogeneous databases makes it possible to develop a single set of portable applications that can be used against both Oracle and IBM databases, and any other databases for which Oracle provides gateways.

Remote Data Access

Location flexibility is maximized because the gateway architecture permits network connections between each of the components. The application can use the Oracle client-server capability to connect to a remote Oracle database through Oracle Net. The Oracle database can connect to a remote gateway using a database link. The gateway connects to a DRDA server through network facilities.

The benefits of remote access are:

- Provides a means to allocate the appropriate resource to a given task
  You can, for example, move application development off expensive processors and onto cost-efficient workstations or microcomputers.
- Expands the number of available data sources
  Without remote access, you are limited to the data available in the local environment. With remote access, only your networks limit your data sources.
- Provides a means to tailor an application environment to a given user
  For example, some users prefer a block-mode terminal environment, while others prefer a bit-mapped, graphics driven terminal environment. Remote access can satisfy both because you are not constrained by the interface environment imposed by the location of your data.

Support for Distributed Applications

Because the gateway gives your application direct access to DRDA data, you eliminate the need to upload and download large quantities of database data to other processors. Instead, you can access data where it is, when you want it, without having...
to move the data between machines and risk unsynchronized and inconsistent data. Avoiding massive data replication can also reduce aggregate disk storage requirements over all your systems.

However, if your system design requires moving data among the machines in a network, SQL*Plus and the gateway can simplify the data transfer. With a single SQL*Plus command, you can move entire sets of data from one node of the network to another and from one database to another.

You can pass commands and statements specific to your DRDA database through the gateway to be executed by the DRDA database. For example, you can pass native DB2 SQL through the gateway for DB2 to execute. You can also execute stored procedures defined in non-Oracle databases.

Application Development and End User Tools

Through the gateway, Oracle extends the range of application development and end-user tools you can use to access your IBM databases. These tools increase application development and user productivity by reducing prototype, development, and maintenance time. Current Oracle database users do not have to learn a new set of tools to access data stored in DRDA databases. Instead, they can access Oracle database and DRDA data with a single set of tools.

With the gateway and the application development tools available from Oracle you can develop a single set of applications to access Oracle database and DRDA data. Users can use the decision support tools available from Oracle to access Oracle database and DRDA data. These tools can run on remote machines connected through Oracle Net to the Oracle database.

When designing applications, keep in mind that the gateway is designed for retrieval and relatively light transaction loads. The gateway is not currently designed to be a heavy transaction processing system.

Password Encryption Utility

Oracle Database Gateway for DRDA includes a utility to support encryption of plain-text passwords in the Gateway Initialization File. Refer to Chapter 15, "Security Considerations" in Oracle Database Gateway Installation and Configuration Guide for IBM AIX on POWER Systems (64-Bit), Linux x86-64, Oracle Solaris on SPARC (64-Bit), Oracle Solaris on x86-64 (64-Bit) and HP-UX Itanium or Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows for details.

Support for DB2 UDB for z/OS Stored Procedures

Oracle Database Gateway for DRDA supports the native stored procedures.

IBM DB2 Universal Database Support

Oracle Database Gateway for DRDA supports IBM DB2 Universal Database.

DB2 z/OS ASCII and UNICODE Table Support

Oracle Database Gateway for DRDA supports EBCDIC, ASCII, and UNICODE table for DB2 z/OS. The character set selection is defined during table creation.
Read-Only Support

The current release enables the gateway to be configured as a read-only gateway. In this mode, the user will not be able to modify data or call remote procedures at the DRDA database.

Support for Graphic and Multi-byte Data

The current release of the gateway adds support for DB2 GRAPHIC and VARGRAPHIC data types. Refer to Developing Applications for more information.

Support for DB2 Universal Database on Intel Hardware

The current release of the gateway adds support for DRDA servers running on Microsoft Windows and Linux on Intel hardware.

Data Dictionary Support for DB2 Universal Database

The current release of the gateway also provides Oracle data dictionary support for DB2 UDB.
Release Information

The following sections describe release information specific to the Oracle Database Gateway for DRDA.

- Product Set
- Changes and Enhancements
- Product Migration
- Known Problems
- Known Restrictions

2.1 Product Set

The following product components are included in the product installation media:

- Oracle Database Gateway for DRDA, 12c Release 2 (12.2)
- Oracle Net, 12c Release 2 (12.2)

2.2 Changes and Enhancements

The following sections describe the changes and enhancements unique to this release of the gateway:

- Remote Insert Rowsource
- Gateway Password Encryption Tool
- Result Sets and Stored Procedures

2.2.1 Remote Insert Rowsouce

By Oracle Database design, some distributed statement must be executed at the database link site. But in certain circumstances, there is data needed to execute these queries that must be fetched from the originating Oracle Database. Under homogeneous connections, the remote Oracle database would call back the source Oracle database for such data. But in heterogeneous connections, this is not viable, as this means that the Foreign Data Store would have to query call back functions, or data, that can only be provided by the Oracle instance that issued the query. In general, these kinds of statements are not something that can be supported through the Oracle Database Gateway.

The following categories of SQL statements results in a callback:

- Any DML with a sub-select, which refers to a table in Oracle database.
• 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 involve the selection of data from the originating Oracle database.

• Any SQL statement that involves a table in Oracle database, and a LONG or LOB column in a remote table.

A new remote insert rowsource feature has been added to allow remote insert requiring local oracle data to work through the Oracle database and Oracle Database Gateway. This functionality is new, and requires the Oracle database, and the Oracle Database Gateway to be version 12.2 or newer.

An example of a remote INSERT statement that can work through the remote insert rowsource feature is as follows:

```
INSERT INTO gateway_table@gateway_link select * from local_table;
```

### 2.2.2 Gateway Password Encryption Tool

The Gateway Password Encryption tool (g4drpwd) has been replaced by a generic feature that is now part of Heterogeneous Services. Refer to Chapter 15, "Security Considerations" in Oracle Database Gateway Installation and Configuration Guide for IBM AIX on POWER Systems (64-Bit), Linux x86-64, Oracle Solaris on SPARC (64-Bit), Oracle Solaris on x86-64 (64-Bit) and HP-UX Itanium or Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows for details.

### 2.2.3 Result Sets and Stored Procedures

The Oracle Database Gateway for DRDA provides support for stored procedures that 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 \texttt{HS\_FDS\_RESULTSET\_SUPPORT} parameter in the initialization parameter file.

**See Also:**

 Initialization Parameters for information about editing the initialization parameter file and the \texttt{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 \texttt{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 DRDA, you will be in the sequential mode of Heterogeneous Services.
The gateway 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 as follows:

- 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 retrieved or not.

In the following example, the UDB stored procedure is executed to fetch the contents of the `EMP` and `DEPT` tables from UDB:

```sql
CREATE PROCEDURE REFCURPROC (IN STRIN VARCHAR(255), OUT STROUT VARCHAR(255) ) RESULT SETS 3 LANGUAGE SQL
BEGIN
    DECLARE TEMP CHAR (20);
    DECLARE C1 CURSOR WITH RETURN TO CALLER FOR
    SELECT * FROM TKHOEMP;
    DECLARE C2 CURSOR WITH RETURN TO CALLER FOR
    SELECT * FROM TKHODEPT;
    OPEN C1;
    OPEN C2;
    SET STROUT = STRIN;
END
```

### 2.3 Product Migration

Refer to [Oracle Database Gateway Installation and Configuration Guide for IBM AIX on POWER Systems (64-Bit), Linux x86-64, Oracle Solaris on SPARC (64-Bit), Oracle Solaris on x86-64 (64-Bit) and HP-UX Itanium or Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows](#) for information on migrating product configurations from previous releases for additional changes or requirements.

### 2.4 Known Problems

The problems that are documented in the following section are specific to the Oracle Database Gateway for DRDA, and are known to exist in this release of the product. If you have any questions or concerns about these problems, contact Oracle Support Services.

A current list of problems is available online. Contact your local Oracle office for information about accessing this online information.
2.5 Known Restrictions

The following restrictions are known to exist for the products in the 12c Release 2 (12.2). Restrictions are not scheduled to change in future releases. Refer to Developing Applications, for information or limitations when developing your applications.

2.5.1 DB2 Considerations

The following considerations exist in the 12c Release 2 (12.2):

2.5.1.1 DD Basic Tables and Views

The owner of DD basic tables and views is OTGDB2. This cannot be changed.

2.5.1.2 SUBSTR Function Post-Processed

The SUBSTR function can be used with the Oracle database in ways that are not compatible with a DRDA server, such as DB2 UDB for z/OS. Therefore, the SUBSTR function is post-processed. However, it is possible to allow the server to process it natively using the "Native Semantics" feature. Refer to Developing Applications, for details.

2.5.1.3 Data type Limitations

Refer to "DRDA Data type to Oracle Data type Conversion" for detailed information about data types.

2.5.1.4 Null Values and Stored Procedures

Null values are not passed into, or returned from, calls to stored procedures through the gateway.

2.5.1.5 String Concatenation of Numbers

DB2 Universal Database does not support string concatenation of numbers. For example,

```sql
SELECT 2||2 FROM table@dblink
```

is not allowed.

2.5.1.6 GLOBAL_NAMES Initialization Parameter

If GLOBAL_NAMES is set to TRUE in the Oracle database INIT.ORA file, then in order to be able to connect to the gateway, you must specify the Heterogeneous Services (HS) initialization parameter, HS_DB_DOMAIN, in the Gateway Initialization Parameter file to match the value of the DB_DOMAIN parameter of the Oracle database. Refer to Oracle Database Gateway Installation and Configuration Guide for IBM AIX on POWER Systems (64-Bit), Linux x86-64, Oracle Solaris on SPARC (64-Bit), Oracle Solaris on x86-64 (64-Bit) and HP-UX Itanium or Oracle Database Gateway Installation and
2.5.1.7 DRDA Package and DB2 considerations

The gateway utilizes a package for statement execution. This package will be implicitly bound upon the first time the gateway connects to the target DB2 system. Ensure that the user ID connecting to the DB2 system has the necessary privileges to bind a package. Refer to Oracle Database Gateway Installation and Configuration Guide for IBM AIX on POWER Systems (64-Bit), Linux x86-64, Oracle Solaris on SPARC (64-Bit), Oracle Solaris on x86-64 (64-Bit) and HP-UX Itanium or Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows, depending on your platform, for more information.

2.5.1.8 Date Arithmetic

In general, the following types of SQL expression forms do not work correctly with the gateway because of DB2 limitations:

- `date + number`
- `number + date`
- `date - number`
- `date1 - date2`

DB2 does not allow number addition or subtraction with date data types. The date and number addition and subtraction forms are sent through to the DB2 where they are rejected.

Also, DB2 does not perform date subtraction consistently. When you subtract two dates `(date1 - date2)`, differing interpretations of date subtraction in DB2 cause the results to vary by server.

**Note:**

Avoid date arithmetic expressions in all gateway SQL expressions until date arithmetic problems are resolved.

2.5.1.9 Row Length Limitation

Because of a restriction of the DRDA architecture, rows with aggregate length exceeding 32 KB in DRDA representation cannot be stored or retrieved.

2.5.1.10 LONG Data type in SQL*Plus

SQL*Plus cannot fetch LONG columns from the Oracle Database Gateway for DRDA.

2.5.1.11 Stored Procedures and Transaction Integrity

IBM DB2 has introduced a feature called Commit on Return for stored procedures. This feature allows DB2 to perform an automatic commit after a stored procedure runs successfully. This feature is enabled when the procedure is created. To ensure data integrity, the Oracle Database Gateway for DRDA does not support this feature in a
heterogeneous environment. When attempting to call a stored procedure that has this feature enabled, through the gateway, the gateway will return an error, ORA-28526 or PLS-00201 (identifier must be declared).

2.5.2 SQL Limitations

The SQL limitations for Oracle Database Gateway for DRDA are described in the following sections:

2.5.2.1 Oracle ROWID Column

The DB2 ROWID column is not compatible with the Oracle ROWID column. Because the ROWID column is not supported, the following restrictions apply:

- UPDATE and DELETE are not supported with the WHERE CURRENT OF CURSOR clause. To update or delete a specific row through the gateway, a condition style WHERE clause must be used. (Bug No. 205538)

  When UPDATE and DELETE statements are used in precompiler and PL/SQL programs, they rely internally on the Oracle ROWID function.

- Snapshots between Oracle database and DB2 are not supported.

  Snapshots rely internally on the Oracle ROWID column.

2.5.2.2 Oracle Bind Variables

Oracle bind variables become SQL parameter markers when used with the gateway. Therefore, the bind variables are subject to the same restrictions as SQL parameter markers.

For example, the following statements are not allowed:

```sql
WHERE :x IS NULL
WHERE :x = :y
```

2.5.2.3 CONNECT BY Is Not Supported

Oracle Database Gateway for DRDA does not support CONNECT BY in SELECT statements.
Using the Oracle Database Gateway for DRDA

Using the Oracle Database Gateway for DRDA involves connecting to the corresponding gateway system and the remote DRDA database associated with the gateway.

- DRDA Gateway Features
- Processing a Database Link
- Accessing the Gateway
- Accessing i5/OS File Members
- Using the Synonym Feature
- Performing Distributed Queries
- Replicating in a Heterogeneous Environment
- Copying Data from Oracle Database to DRDA Server
- Copying Data from DRDA Server to Oracle Database
- Tracing SQL Statements

3.1 DRDA Gateway Features

This section describes the following DRDA gateway features:

- CHAR Semantics
- Multi-byte Character Sets Ratio Suppression
- IPv6 Support
- Gateway Session IDLE Timeout

3.1.1 CHAR Semantics

This feature allows the gateway to optionally run in CHAR Semantics mode. Rather than always describing UDB CHAR columns as \texttt{CHAR(n \ BYTE)}, this feature describes them as \texttt{CHAR(n \ CHAR)} and \texttt{VARCHAR(n \ CHAR)}. The concept is similar to Oracle database CHAR Semantics. You need to specify \texttt{HS\_NLS\_LENGTH\_SEMANTICS=CHAR} gateway parameter to activate this option. Refer to \texttt{Initialization Parameters} for more detail.

3.1.2 Multi-byte Character Sets Ratio Suppression

This feature optionally suppresses the ratio expansion from UDB database to Oracle database involving multi-byte character set. By default, Oracle Database Gateway for DRDA assumes the worst ratio to prevent data being truncated or insufficient buffer size situation. However, if you have specific knowledge of your UDB database and do
not want the expansion to occur, you can specify `HS_KEEP_REMOTE_COLUMN_SIZE` parameter to suppress the expansion. Refer to Initialization Parameters for more detail.

### 3.1.3 IPv6 Support

Besides full IPv6 support between Oracle databases and the gateway, IPv6 is also supported between this gateway and UDB database. Refer to the `HS_FDS_CONNECT_INFO` parameter in Initialization Parameters for more detail.

### 3.1.4 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 update rolled back.

### 3.2 Processing a Database Link

The database and application administrators of a distributed database system are responsible for managing the database links that define paths to the Oracle Database Gateway for DRDA. The tasks are as follows:

- Creating Database Links
- Dropping Database Links
- Examining Available Database Links
- Limiting the Number of Active Database Links

#### 3.2.1 Creating Database Links

To create a database link and define a path to a remote database, use the `CREATE DATABASE LINK` statement. The `CONNECT TO` clause specifies the remote user ID and password to use when creating a session in the remote database. The `USING` clause points to a tnsnames.ora connect descriptor.

**Note:**

If you do not specify a user ID and a password in the `CONNECT TO` clause, then the Oracle database user ID and password are used.

**See Also:**

Refer to Chapter 15, "Security Considerations" in Oracle Database Gateway Installation and Configuration Guide for IBM AIX on POWER Systems (64-Bit), Linux x86-64, Oracle Solaris on SPARC (64-Bit), Oracle Solaris on x86-64 (64-Bit) and HP-UX Itanium or Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows for details.
The following example creates a database link to access information in the DRDA server:

```sql
CREATE PUBLIC DATABASE LINK dblink
CONNECT TO userid IDENTIFIED BY password
USING 'tns_name_entry';
```

where:

dblink is the complete database link name.

userid is the user ID used to establish a session in the remote database. This user ID must be a valid DRDA server user ID. It must be authorized to any table or file on the DRDA server that is referenced in the SQL commands. Length restrictions on user IDs are dependent on the authorization system used by the DRDA server. In many cases this limit is eight characters, but in other cases, it may be longer. See DB2 platform documentation for limitations.

password is the password used to establish a session in the remote database. This password must be a valid DRDA server password. Length restrictions on passwords are dependent on the authorization system used by the DRDA server. In many cases this limit is eight characters, but in other cases, it may be longer. See DB2 platform documentation for limitations.

tns_name_entry specifies the Oracle Net connect descriptor used to identify the gateway.

### Guidelines for Database Links

Database links are active for the duration of a gateway session. If you want to close a database link during a session, then use the `ALTER SESSION CLOSE DATABASE LINK dblink` statement.

### 3.2.2 Dropping Database Links

You can drop a database link with the `DROP DATABASE LINK` statement. For example, to drop the public database link named `DBLINK`, use the statement:

```sql
DROP PUBLIC DATABASE LINK dblink;
```

---

**Note:**

A database link should not be dropped if it is required to resolve an in-doubt distributed transaction. Refer to *Oracle Database Administrator's Guide* for additional information about dropping database links.

**See Also:**

*Oracle Database SQL Language Reference* for additional information about dropping database links
3.2.3 Examining Available Database Links

The data dictionary of each database stores the definitions of all the database links in that database. The USER_DB_LINKS data dictionary view shows the privately defined database links, that is, those accessible to the current Oracle user. The ALL_DB_LINKS data dictionary views show all accessible (public and private) database links.

3.2.4 Limiting the Number of Active Database Links

You can limit the number of connections from a user process to remote databases with the parameter OPEN_LINKS. This parameter controls the number of remote connections that any single user process can concurrently use with a single SQL statement. Refer to Oracle Database Administrator’s Guide for additional information about limiting the number of active database links.

3.3 Accessing the Gateway

To access the gateway, complete the following steps on the Oracle database:

1. Example 3-1
2. Example 3-2
3. Example 3-3

Example 3-1   Login to the Oracle Database
Login to the Oracle database to access the gateway

Example 3-2   Create a database link to the DRDA Database
For example, use:

CREATE PUBLIC DATABASE LINK DRDA
CONNECT TO ORADRDA IDENTIFIED BY oracle_pw
USING 'tns_name_entry'

Example 3-3   Retrieve data from the DRDA Database
This query fetches the TABLE table in the schema SECURE, using the name ORADRDA as the DRDA server user profile. The ORADRDA user profile must have the appropriate privileges on the DRDA server to access the SECURE.TABLE files:

SELECT * FROM SECURE.TABLE@DRDA

The following is an example of the error messages that are displayed if insufficient privileges are displayed:

ORA-28500: connection from ORACLE to a non-Oracle system returned this message: ORADRDA DOES NOT HAVE PRIVILEGE TO PERFORM OPERATION SELECT ON THIS OBJECT SECURE.TABLE.
ORA-02063: preceding 2 lines from DRDA
3.4 Accessing i5/OS File Members

Access to i5/OS files and file members is not specifically controlled by DRDA or the gateway. However, DB2 UDB for iSeries uses a naming convention that implies that the file member name is the same as the name of the file being addressed. For example, accessing `schema.table` implies that `table` is the file name and also that `table` is the file member name being accessed.

To access file members with names that differ from the associated file name, you must create a view within the file so that DB2 UDB for iSeries can reference the correct file member.

One method for creating this view involves issuing the i5/OS command `CRTLF` (Create Logical File). This action creates a logical association between the file name and the file member name.

See Also:

For additional information, refer to the i5/OS Command Language (CL) documentation or to the DB2 UDB for iSeries SQL reference document.

3.5 Using the Synonym Feature

You can provide complete data, location, and network transparency by using the synonym feature of Oracle database. When a synonym is defined, the user need not know the underlying table or network protocol being used. A synonym can be public, which means it is available to all Oracle users. A synonym can also be defined as private, available only to the user who created it. Refer to Oracle Database Reference for details on the synonym feature.

The following statement creates a system-wide synonym for the `EMP` file in the DRDA server with ownership of ORACLE:

```
CREATE PUBLIC SYNONYM EMP FOR ORACLE.EMP@DRDA
```

3.6 Performing Distributed Queries

The Oracle Database Gateway technology enables the execution of distributed queries that join data in an Oracle database and in DRDA servers and data from any other data store for which Oracle provides a gateway. These complex operations can be completely transparent to the users requesting the data.

The following example joins data between an Oracle database, DB2 UDB for z/OS, and a DRDA server:

```
SELECT o.custname, p.projno, e.ename, sum(e.rate*p.hours)
FROM orders@DB2 o, EMP@ORACLE7 e, projects@DRDA p
WHERE o.projno = p.projno
AND p.empno = e.empno
GROUP BY o.custname, p.projno, e.ename
```
A combination of views and synonyms, using the following SQL statements, keeps the process of distributed queries transparent to the user:

CREATE SYNONYM orders for orders@DB2;
CREATE SYNONYM PROJECTS for PROJECTS@ORDA;
CREATE VIEW details (custname,projno,ename,spend) AS
SELECT o.custname, p.projno, e.ename, sum(e.rate*p.hours)
FROM orders o, EMP e, projects p
WHERE o.projno = p.projno
AND p.empno = e.empno
GROUP BY o.custname, p.projno, e.ename

The following SQL statement retrieves information from these three data stores in one command:

SELECT * FROM DETAILS;

The results of this command are:

<table>
<thead>
<tr>
<th>CUSTNAME</th>
<th>PROJNO</th>
<th>ENAME</th>
<th>SPEND</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC Co.</td>
<td>1</td>
<td>Jones</td>
<td>400</td>
</tr>
<tr>
<td>ABC Co.</td>
<td>1</td>
<td>Smith</td>
<td>180</td>
</tr>
<tr>
<td>XYZ Inc.</td>
<td>2</td>
<td>Jones</td>
<td>400</td>
</tr>
<tr>
<td>XYZ Inc.</td>
<td>2</td>
<td>Smith</td>
<td>180</td>
</tr>
</tbody>
</table>

3.6.1 Two-Phase Commit Processing

To fully participate in a two-phase commit transaction, a server must support the PREPARE TRANSACTION statement. The PREPARE TRANSACTION statement ensures that all participating databases are prepared to COMMIT or to ROLLBACK a specific unit of work.

Oracle database supports the PREPARE TRANSACTION statement. Any number of Oracle database can participate in a distributed two-phase commit transaction. The PREPARE TRANSACTION statement is performed automatically when a COMMIT is issued explicitly by an application or implicitly at the normal end of the application.

The gateway does not support the PREPARE TRANSACTION statement. This limits the two-phase commit protocol when the gateway participates in a distributed transaction. The gateway becomes the commit focal point site of a distributed transaction. The gateway is configured as commit/confirm, so it is always the commit point site, regardless of the commit point strength setting. The gateway commits the unit of work after verifying that all Oracle databases in the transaction have successfully committed their work. The gateway must coordinate the distributed transaction so that only one gateway instance can participate in a two-phase commit transaction.

Two-phase commit transactions are recorded in the HS_TRANSACTION_LOG table (see the initialization parameter HS_FDS_TRANSACTION_LOG), which is created during installation. This table is created when the o2pc.sql script is run. The owner of this table also owns the package. Refer to “DRDA Gateway Package Binding Considerations” on Oracle Database Gateway Installation and Configuration Guide for IBM AIX on POWER Systems (64-Bit), Linux x86-64, Oracle Solaris on SPARC (64-Bit), Oracle Solaris on x86-64 (64-Bit) and HP-UX Itanium or Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows, depending on your platform, for more information.
3.6.2 Distributed DRDA Transactions

Because the `HS_TRANSACTION_LOG` table is used to record the status of a gateway transaction, this table must be in at the database where the DRDA update takes place. Therefore, all updates that take place over the gateway must be local to the IBM database.

**Note:**

- Updates to the `HS_TRANSACTION_LOG` table cannot be part of an IBM distributed transaction.
- The default commit mode on OS400 V5R1 and later is `READ UNCOMMITTED (*CHG)` and this requires files to be journaled. Hence, the object specified by the `HS_TRANSACTION_LOG` initialization parameter must be journaled.

For additional information about the two-phase commit process, refer to *Oracle Database Heterogeneous Connectivity User's Guide*.

3.7 Replicating in a Heterogeneous Environment

Oracle Database Gateway for DRDA provides a number of options for replicating Oracle and non-Oracle data throughout the enterprise.

**Oracle Database Triggers**

When updates are made to Oracle database, synchronous copies of Oracle and non-Oracle data can be maintained automatically by using Oracle database triggers.

**Oracle Snapshots**

Oracle Database Gateway for DRDA can use the Oracle snapshot feature to automatically replicate non-Oracle data into Oracle database. The complete refresh capability of Oracle snapshot can be used to propagate a complete copy or a subset of the non-Oracle data into Oracle database at user-defined intervals.

3.8 Copying Data from Oracle Database to DRDA Server

This Oracle SQL INSERT command now works with the CALLBACK feature:

```
INSERT INTO DRDA_table SELECT * FROM local_table
```

Alternatively, you could use sqlplus `COPY` command. The `COPY` command enables you to copy data from Oracle database to a DRDA server. To copy data from your Oracle database to the DRDA server, you can also use:

```
COPY FROM username/password@connect_identifier -
INSERT destination_table -
USING query
```
For example, to select all rows from the local Oracle EMP table, insert them into the EMP table on the DRDA server, and commit the transaction, use:

COPY FROM scott/tiger@ORACLE -
INSERT scott.EMP@DRDA -
USING SELECT * FROM EMP

The SQL*Plus COPY command supports APPEND, CREATE, INSERT, and REPLACE commands. However, INSERT is the only command supported when copying to the DRDA server. For more information about the COPY command, refer to SQL*Plus User’s Guide and Reference.

3.9 Copying Data from DRDA Server to Oracle Database

The CREATE TABLE command enables you to copy data from a DRDA server to Oracle database. To create a table on your Oracle database and to insert rows from a DRDA server table, use:

CREATE TABLE table_name
AS query

A SQL INSERT into an Oracle table can be done selecting data from the gateway as follows:

INSERT INTO local_table SELECT * FROM drda_table

The following example creates the table EMP in your local Oracle database and inserts the rows from the EMP table on the DRDA server:

CREATE TABLE EMP
AS SELECT * FROM scott.EMP@DRDA

Alternatively, you can use the SQL*Plus COPY command to copy data from a DRDA server to Oracle database. For more information about the COPY command, refer to SQL*Plus User’s Guide and Reference.

3.10 Tracing SQL Statements

SQL statements issued through the gateway can be changed before reaching the DRDA database. These changes are made to make the format acceptable to the gateway or to make Oracle SQL compatible with DRDA server SQL. Oracle database and the gateway can change the statements depending on the situation.

For various reasons, you might need to assess whether the gateway has altered the statement correctly or whether the statement could be rewritten to improve performance. SQL tracing is a feature that allows you to view the changes made to a SQL statement by the Oracle database or the gateway.

SQL tracing reduces gateway performance. Use tracing only when testing and debugging your application. Do not enable SQL tracing when the application is running in a production environment. For more information about enabling SQL tracing, refer to the section on "SQL Tracing and the Gateway " in Error Messages, Diagnosis, and Reporting.
Developing Applications

The following sections provide information that is specific to the Oracle Database Gateway for DRDA.

- Gateway Appearance to Application Programs
- Using Oracle Stored Procedures with the Gateway
- Using DRDA Server Stored Procedures with the Gateway
- Database Link Behavior
- Oracle Database SQL Construct Processing
- Native Semantics
- DRDA Data type to Oracle Data type Conversion
- Passing Native SQL Statements through the Gateway
- Oracle Data Dictionary Emulation on a DRDA Server

4.1 Gateway Appearance to Application Programs

An application that is written to access information in a DRDA database interfaces with an Oracle database. When developing applications, keep the following information in mind:

- You must define the DRDA database to the application by using a database link that is defined in the Oracle database. Your application should specify tables that exist on a DRDA database by using the name that is defined in the database link. For example, assume that a database link is defined so that it names the DRDA database link `DRDA`, and also assume that an application needs to retrieve data from an Oracle database and from the DRDA database. Use the following SQL statement joining two tables together in your application:

  ```sql
  SELECT EMPNO, SALARY
  FROM EMP L, EMPS@DRDA R
  WHERE L.EMPNO = R.EMPNO
  ```

  In this example, `EMP` is a table on an Oracle database, and `EMPS` is a table on a DRDA server. You can also define a synonym or a view on the DRDA server table, and access the information without the database link suffix.

- You can read and write data to a defined DRDA database. `SELECT`, `INSERT`, `UPDATE`, and `DELETE` are all valid operations.

- A single transaction can write to one DRDA database and to multiple Oracle databases.

- Single SQL statements, using `JOINs`, can refer to tables in multiple Oracle databases, in multiple DRDA databases, or in both.
4.1.1 Fetch Reblocking

Oracle database supports fetch reblocking with the HS_RPC_FETCH_REBLOCKING parameter.

When the value of this parameter is set to ON (the default), the array size for SELECT statements is determined by the HS_RPC_FETCH_SIZE value. The HS_RPC_FETCH_SIZE parameter defines the number of bytes sent with each buffer from the gateway to the Oracle database. The buffer may contain one or more qualified rows from the DRDA server. This feature can provide significant performance enhancements, depending on your application design, installation type, and workload.

The array size between the client and the Oracle database is determined by the Oracle application. Refer to Oracle Database Gateway Installation and Configuration Guide for IBM AIX on POWER Systems (64-Bit), Linux x86-64, Oracle Solaris on SPARC (64-Bit), Oracle Solaris on x86-64 (64-Bit) and HP-UX Itanium or Oracle Database Gateway Installation and Configuration Guide for Microsoft Windows, depending on your platform, for more information.

4.2 Using Oracle Stored Procedures with the Gateway

The gateway stored procedure support is an extension of Oracle stored procedures. An Oracle stored procedure is a schema object that logically groups together a set of SQL and other PL/SQL programming language statements to perform a specific task. Oracle stored procedures are stored in the database for continued use. Applications use standard Oracle PL/SQL to call stored procedures.

Oracle stored procedures can be located in a local instance of Oracle database and in a remote instance. Figure 4-1 illustrates two stored procedures: oraproc1 is a procedure stored in the ORA1 Oracle instance, and oraproc2 is a procedure stored in the ORA2 Oracle instance.
To maintain location transparency in the application, a synonym can be created:

CREATE SYNONYM oraproc2 FOR oraproc2@ora2;

After this synonym is created, the application no longer needs to use the database link specification to call the stored procedure in the remote Oracle instance.

In Figure 4-1, the second statement in oraproc1 is used to access a table in the ORA2 instance. In the same way, Oracle stored procedures can be used to access DB2 tables through the gateway.

4.3 Using DRDA Server Stored Procedures with the Gateway

The procedural feature of the gateway enables invocation of native DRDA server stored procedures. After the stored procedure is defined to the DRDA server, the gateway is able to use the existing DRDA server definition to run the procedure. The gateway does not require special definitions to call the DB2 stored procedure. Standard Oracle PL/SQL is used by the Oracle application to run the stored procedure.
In Figure 4-2, an Oracle application calls the *empproc* stored procedure that is defined to the DRDA server (for example, DB2 UDB for z/OS).

**Figure 4-2 Running DRDA Server Stored Procedures**

From the perspective of the application, running the DB2 stored procedure is no different from invoking a stored procedure at a remote Oracle database instance.

### 4.3.1 Oracle Application and DRDA Server Stored Procedure Completion

For an Oracle application to call a DB2 stored procedure, it is first necessary to create the DB2 stored procedure on the DB2 system by using the procedures documented in the IBM reference document for DB2 SQL.

After the stored procedure is defined in DB2, the gateway is able to access the data using a standard PL/SQL call. For example, an employee name, John Smythe, is passed to the DB2 stored procedure *REVISE_SALARY*. The DB2 stored procedure retrieves the salary value from the DB2 database in order to calculate a new yearly salary for John Smythe. The revised salary that is returned as result is used to update the EMP table of Oracle database:

```sql
DECLARE
  INPUT VARCHAR2(15);
  RESULT NUMBER(8,2);
BEGIN
  INPUT := 'JOHN SMYTHE';
  REVISE_SALARY@DB2(INPUT, RESULT);
END;
```
UPDATE EMP SET SAL = RESULT WHERE ENAME = INPUT;
END;

When the gateway receives a call to run a stored procedure on the DRDA server, it first does a lookup of the procedure name in the server catalog. The information that defines a stored procedure is stored in different forms on each DRDA server. For example, DB2 UDB for iSeries uses the tables QSYS2.SYSPROCS and QSYS2.SYSPARMS. The gateway has a list of known catalogs to search, depending on the DRDA server that is being accessed.

The search order of the catalogs is dependent on whether the catalogs support Location designators (such as LUNAME in SYSIBM.SYSPROCEDURES), and authorization or owner IDs (such as AUTHID in SYSIBM.SYSPROCEDURES or OWNER in SYSIBM.SYSROUTINES).

Some DRDA servers allow blank or public authorization qualifiers. If the DRDA server that is currently connected supports this form of qualification, then the gateway will apply those naming rules when searching for a procedure name in the catalog.

The matching rules will first search for a public definition, and then an owner qualified procedure name. For more detailed information, refer to the IBM reference document for DB2 SQL.

4.3.2 Procedural Feature Considerations with DB2

The following are special considerations for using the procedural feature with the gateway:

- PL/SQL records cannot be passed as parameters when invoking a DB2 stored procedure.
- The gateway supports the GENERAL and DB2SQL linkage conventions of DB2 stored procedures. Both linkage conventions require that the parameters that are passed to and from the DB2 stored procedure cannot be null.

4.3.3 Result Sets and Stored Procedures

The Oracle Database Gateway for DRDA provides support for stored procedures that 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:

Initialization Parameters for information about 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.
When accessing stored procedures with result sets through the Oracle Database Gateway for DRDA, you will be in the sequential mode of Heterogeneous Services. The gateway 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 as follows:

- 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 retrieved or not.

In the following example, the UDB stored procedure is executed to fetch the contents of the EMP and DEPT tables from UDB:

```sql
CREATE PROCEDURE REFCURPROC (IN STRIN VARCHAR(255), OUT STROUT VARCHAR(255) ) 
RESULT SETS 3  LANGUAGE SQL
BEGIN
  DECLARE TEMP CHAR (20);
  DECLARE C1 CURSOR WITH RETURN TO CALLER FOR
    SELECT * FROM TKHOEMP;
  DECLARE C2 CURSOR WITH RETURN TO CALLER FOR
    SELECT * FROM TKHODEPT;
  OPEN C1;
  OPEN C2;
  SET STROUT = STRIN;
END
```

### 4.3.3.1 OCI Program Fetching from Result Sets in Sequential Mode

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

```sql
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@UDB(:1,:2,:3); end;";
OraText *n_rs_stm = (OraText *)
  "begin :ret := DBMS_HS_RESULT_SET.GET_NEXT_RESULT_SET@UDB; 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 */

4.3.3.2 PL/SQL Program Fetching from Result Sets in Sequential Mode

Assume that the table LOC_EMP is a local table exactly like the UDB EMP table. The
same assumption applies for LOC_DEPT. OUTARGS is a table with columns corresponding
to the out arguments of the SQL Server stored procedure.

create or replace package rcpackage is  type RCTYPE is ref cursor;end rcpackage;/
declare
   rcl 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@UDB('Hello World', out_arg, rcl);

   -- Fetch 20 rows from the remote emp table and insert them into loc_emp
   for i in 1 .. 20 loop
     fetch rcl 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 rcl;

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

   -- 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@UDB;

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

A connection to the gateway is established through a database link when it is first used in an Oracle database session. In this context, a connection refers to both the connection between the Oracle database and the gateway and to the DRDA network connection between the gateway and the target DRDA database. The connection remains established until the Oracle database session ends. Another session or user can access the same database link and get a distinct connection to the gateway and DRDA database.

4.5 Oracle Database SQL Construct Processing

One of the most important features of the Oracle Database Gateways products is providing SQL transparency to the user and to the application programmer. Foreign SQL constructs can be categorized into four areas:

- Compatible
- Translated
- Compensated
- Native semantics

4.5.1 Compatible SQL Functions

Oracle database automatically forwards compatible SQL functions to the DRDA database, where SQL constructs with the same syntax and meaning are on both Oracle database and the DRDA database. These SQL constructs are forwarded unmodified. All of the compatible functions are column functions. Functions that are not compatible are either translated to an equivalent DRDA SQL function or are compensated (post-processed) by Oracle database after the data is returned from the DRDA database.

4.5.2 Translated SQL Functions

Translated functions have the same meaning but different names between the Oracle database and the DRDA database. But all applications must use the Oracle function name. These SQL constructs that are supported with different syntax (different function names) by the DRDA database, are automatically translated by the Oracle database and then forwarded to the DRDA database. Oracle database changes the function name before sending it to the DRDA database, in a manner that is transparent to your application.

4.5.3 Compensated SQL Functions

Some advanced SQL constructs that are supported by Oracle database may not be supported in the same manner, by the DRDA database. Compensated functions are those SQL functions that are either not recognized by the DRDA server or are
recognized by the DRDA server but the semantics of the function are interpreted differently when comparing the DRDA server with the Oracle database. If a SELECT statement containing one of these functions is passed from the Oracle database to the gateway, then the gateway removes the function before passing the SQL statement to the DRDA server. The gateway passes the selected DRDA database rows to Oracle database. Oracle database applies the function.

4.5.3.1 Post-Processing

Oracle database can compensate for a missing or incompatible function by automatically excluding the incompatible SQL construct from the SQL request that is forwarded to the DRDA database. Oracle database then retrieves the necessary data from the DRDA database and applies the function. This process is known as post-processing.

The gateway attempts to pass all SQL functions to DRDA databases. However when a DRDA database does not support a function that is represented in the computation, the gateway changes that function. For example, if a program runs the following query against a DB2 UDB for z/OS database:

```
SELECT COS(X_COOR) FROM TABLE_X;
```

Because the database does not support many of the \texttt{COS} functions, the gateway changes the query to the following:

```
SELECT X_COOR FROM TABLE_X;
```

All data in the \texttt{X_COOR} column of \texttt{TABLE_X} is passed from the DB2 UDB for z/OS database to the Oracle database. After the data is moved to the Oracle database, the \texttt{COS} function is performed.

If you are performing operations on large amounts of data that are stored in a DRDA database, then keep in mind that some functions require post-processing.

4.5.4 Native Semantic SQL Functions

Some SQL functions that are normally compensated may also be overridden, through the Native Semantics facility. If a SQL function has been enabled for Native Semantics, then the function may be passed on to the DRDA database for processing, instead of being compensated. The SQL function is then processed natively in the DRDA database. Refer to "Native Semantics" for more information.

4.5.5 DB2 UDB for z/OS SQL Compatibility

\textbf{Table 4-1} describes how Oracle database and the gateway handle SQL functions for a DB2 UDB for z/OS.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{Oracle SQL Function} & \textbf{Compatible} & \textbf{Translated} & \textbf{Compensated} & \textbf{Native Semantics Candidate} \\
\hline
\texttt{ABS} & - & - & Yes & Yes \\
\texttt{ACOS} & - & - & Yes & Yes \\
\hline
\end{tabular}
\end{table}
### Table 4-1  (Cont.) SQL Compatibility, by Oracle SQL function

<table>
<thead>
<tr>
<th>Oracle SQL Function</th>
<th>Compatible</th>
<th>Translated</th>
<th>Compensated</th>
<th>Native Semantics Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD_MONTHS</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>ASCII</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ASIN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ATAN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ATAN2</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AVG</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>BITAND</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CAST</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CEIL</td>
<td>-</td>
<td>CEILING</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>CHARTOROWID</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>CHR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CONCAT</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>CONVERT</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COS</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COSH</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COUNT(*)</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>COUNT (DISTINCT colname)</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>COUNT (ALL colname)</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>COUNTCOL</td>
</tr>
<tr>
<td>COUNT (column)</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>COUNTCOL</td>
</tr>
<tr>
<td>DECODE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DUMP</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EXP</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FLOOR</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>GREATEST</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HEXTORAW</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INITCAP</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INSTR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INSTRB</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LAST_DAY</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>LEAST</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LENGTH</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LENGTHB</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOG</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOWER</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>LCASE</td>
</tr>
</tbody>
</table>
Table 4-1  (Cont.) SQL Compatibility, by Oracle SQL function

<table>
<thead>
<tr>
<th>Oracle SQL Function</th>
<th>Compatible</th>
<th>Translated</th>
<th>Compensated</th>
<th>Native Semantics Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPAD</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LTRIM</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MAX</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MIN</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MOD</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MONTHS_BETWEEN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>NEW_TIME</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>NEXT_DAY</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>NLS_INITCAP</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NLS_LOWER</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NLS_UPPER</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NLSSORT</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NVL</td>
<td>VALUE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NVL2</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>POWER</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAWTOHEX</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPLACE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REVERSE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ROUND</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ROWIDTOCHAR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>RPAD</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RTRIM</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIGN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COS</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SINGH</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SOUNDINDEX</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>SQRT</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>STDDEV</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBSTR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBSTRB</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUM</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SYSDATE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TAN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TANH</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_CHAR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TO_DATE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 4-1  (Cont.) SQL Compatibility, by Oracle SQL Function

<table>
<thead>
<tr>
<th>Oracle SQL Function</th>
<th>Compatible</th>
<th>Translated</th>
<th>Compensated</th>
<th>Native Semantics Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO_MULTI_BYTE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TO_NUMBER</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_SINGLE_BYTE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TRANSLATE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TRIM</td>
<td>-</td>
<td>STRIP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TRUNC</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UID</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>UPPPER</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>UCASE</td>
</tr>
<tr>
<td>USER</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>USERENV</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>VARIANCE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VSIZE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 4.5.6 DB2 UDB for Unix, Linux, and Windows Compatibility

Table 4-2 describes how Oracle database and the gateway handle SQL functions for a DB2/UDB database.

### Table 4-2  DB2 UDB for Unix, Linux, and Windows Compatibility, by Oracle SQL Function

<table>
<thead>
<tr>
<th>Oracle SQL Function</th>
<th>Compatible</th>
<th>Translated</th>
<th>Compensated</th>
<th>Native Semantics Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>ACOS</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ADD_MONTHS</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>ASCII</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ASIN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ATAN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ATAN2</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AVG</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BITAND</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CAST</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CEIL</td>
<td>-</td>
<td>CEILING</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>CHARTOROWID</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>CHR</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>CONCAT</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 4-2  (Cont.) DB2 UDB for Unix, Linux, and Windows Compatibility, by Oracle SQL Function

<table>
<thead>
<tr>
<th>Oracle SQL Function</th>
<th>Compatible</th>
<th>Translated</th>
<th>Compensated</th>
<th>Native Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONVERT</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COS</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>COSH</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COUNT(*)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNT (DISTINCT colname)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNT (ALL colname)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNT (column)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECODE</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DUMP</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EXP</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOOR</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GREATEST</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HEXTORAW</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INITCAP</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INSTR</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INSTRB</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LAST_DAY</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LEAST</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LENGTH</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LENGTHB</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LN</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOG</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LOWER</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LPAD</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>LTRIM</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>MAX</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOD</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MONTHS_BETWEEN</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NEW_TIME</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NEXT_DAY</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NLS_INITCAP</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NLS_LOWER</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NLS_UPPER</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Table 4-2  (Cont.) DB2 UDB for Unix, Linux, and Windows Compatibility, by Oracle SQL Function

<table>
<thead>
<tr>
<th>Oracle SQL Function</th>
<th>Compatible</th>
<th>Translated</th>
<th>Compensated</th>
<th>Native Semantics Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLSSORT</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NVL</td>
<td>-</td>
<td>VALUE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NVL2</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>POWER</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RAWTOHEX</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPLACE</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>REVERSE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ROUND</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ROWIDTOCHAR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>RPAD</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RTRIM</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIGN</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SIN</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SINH</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SOUNDSEX</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>SQRT</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STDDEV</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBSTR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBSTRB</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUM</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SYSDATE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TAN</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TANH</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_CHAR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TO_DATE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TO_MULTI_BYTE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TO_NUMBER</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_SINGLE_BYTE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TRANSLATE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TRIM</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TRUNC</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UID</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>UPPER</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>UCASE</td>
</tr>
<tr>
<td>USER</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>USERNV</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 4-2 (Cont.) DB2 UDB for Unix, Linux, and Windows Compatibility, by Oracle SQL Function

<table>
<thead>
<tr>
<th>Oracle SQL Function</th>
<th>Compatible</th>
<th>Translated</th>
<th>Compensated</th>
<th>Native Semantics Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARIANCE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VSIZE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### 4.5.7 DB2 UDB for iSeries Compatibility

Table 4-3 describes how Oracle database and the gateway handle SQL functions for a DB2 UDB for iSeries database.

### Table 4-3 DB2 UDB for iSeries Compatibility, by Oracle SQL Function

<table>
<thead>
<tr>
<th>Oracle SQL Function</th>
<th>Compatible</th>
<th>Translated</th>
<th>Compensated</th>
<th>Native Semantics Candidate</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>-</td>
<td>ABSVAL</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>ACOS</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ADD_MONTHS</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>ASCII</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ASIN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ATAN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ATANZ</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AVG</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>BITAND</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CAST</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CEIL</td>
<td>-</td>
<td>CEILING</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>CHARTOROWID</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>CHR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CONCAT</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CONVERT</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>COS</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>COSH</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>COUNT(*)</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>COUNT (DISTINCT colname)</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>COUNT (ALL colname)</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>COUNTCOL</td>
</tr>
<tr>
<td>COUNT (column)</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>COUNTCOL</td>
</tr>
<tr>
<td>DECODE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DUMP</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Oracle SQL Function</td>
<td>Compatible</td>
<td>Translated</td>
<td>Compensated</td>
<td>Native Semantics Candidate</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>------------</td>
<td>-------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>EXP</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>FLOOR</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>GREATEST</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>HEXTORAW</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INITCAP</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INSTR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>INSTRB</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LAST_DAY</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>LEAST</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LENGTH</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LENGTHB</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LN</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>LOG</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LOWER</td>
<td>Yes</td>
<td>-</td>
<td>No</td>
<td>LCASE</td>
</tr>
<tr>
<td>LPAD</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>LTRIM</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MAX</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MIN</td>
<td>Yes</td>
<td>-</td>
<td>v</td>
<td>-</td>
</tr>
<tr>
<td>MOD</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>MONTHS_BETWEEN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NEW_TIME</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NEXT_DAX</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>NLS_INITCAP</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NLS_LOWER</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NLS_UPPER</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NLSSORT</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NVL</td>
<td>-</td>
<td>VALUE</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NVL2</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>POWER</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RAWTOHEX</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REPLACE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>REVERSE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ROUND</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ROWIDTOCHAR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>RPAD</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RTRIM</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Oracle SQL Function</td>
<td>Compatible</td>
<td>Translated</td>
<td>Compensated</td>
<td>Native Semantics Candidate</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>SIGN</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SIN</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>SINE</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>SOUNDEX</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>SQRT</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>STDDEV</td>
<td>Yes</td>
<td>v</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBSTR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUBSTRB</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SUM</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SYSDATE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TAN</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>TANH</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_CHAR</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TO_DATE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TO_MULTI_BYTE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TO_NUMBER</td>
<td>-</td>
<td>DECIMAL</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>TO_SINGLE_BYTE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>TRANSLATE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TRIM</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>TRUNC</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UID</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>UPPER</td>
<td>-</td>
<td>TRANSLATE</td>
<td>-</td>
<td>UCASE</td>
</tr>
<tr>
<td>USER</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>USERENV</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>VARIANCE</td>
<td>-</td>
<td>VAR</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>VSIZE</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 4.6 Native Semantics

Some of the advanced SQL constructs that are supported by Oracle database may not be supported in the same manner by the DRDA database. In this case, the Oracle database compensates for the missing or incompatible functionality by post-processing the DRDA database data with Oracle database functionality.
This feature provides maximum transparency, but may impact performance. In addition, new versions of a particular DRDA database may implement previously unsupported functions or capabilities, or they may change the supported semantics as to make them more compatible with Oracle database functions.

Some of DRDA servers also provide support for user-defined functions. The user may choose to implement Oracle database functions natively in the DRDA database. This enables the DRDA server to pass the function to the underlying database implementation (for example, DB2). Native Semantics provides a method of enabling specific capabilities to be processed natively by the DRDA server.

Native Semantics Considerations

Various considerations must be taken into account when enabling the Native Semantic feature of a particular function because Native Semantics has advantages and disadvantages, which are typically a trade-off between transparency and performance.

- One such consideration is the transparency of data coercions. Oracle database provides coercion (implicit data conversion) for many SQL functions. This means that if the supplied value for a particular function is not correct, then Oracle database will coerce the value (change it to the correct value type) before processing it. However, with the Native Semantic feature enabled, the value, exactly as provided, will be passed to the DRDA server for processing. In many cases, the DRDA server will not be able to coerce the value to the correct type and will generate an error.

- Another consideration involves the compatibility of parameters to a particular SQL function. For instance, Oracle database implementation of \textit{SUBSTR} allows negative values for the string index, whereas most DRDA server implementations of \textit{SUBSTR} do not allow negative values for the string index. However, if the application is implemented to invoke \textit{SUBSTR} in a manner that is compatible with the DRDA server, then the function will behave the same in either Oracle database or the DRDA server.

- Another consideration is that the processing of a function at the DRDA server may not be desirable due to resource constraints in that environment.

Refer to the "\textit{HS_FDS_CAPABILITY}" for details on enabling or disabling these capabilities. Refer to the \textit{Oracle Database SQL Language Reference} for Oracle database format of the following capabilities.

4.6.1 SQL Functions That Can Be Enabled

The following list contains SQL functions that are disabled (\textit{OFF}) by default. They can be enabled (turned \textit{ON}) as an option:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>ACOS</td>
</tr>
<tr>
<td>ASCII</td>
<td>ASIN</td>
</tr>
<tr>
<td>Function</td>
<td>Function</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>ATAN</td>
<td>ATAN2</td>
</tr>
<tr>
<td>BITAND</td>
<td>CAST</td>
</tr>
<tr>
<td>CEIL</td>
<td>CHR</td>
</tr>
<tr>
<td>CONVERT</td>
<td>COS</td>
</tr>
<tr>
<td>COSH</td>
<td>COUNTCOL</td>
</tr>
<tr>
<td>DECODE</td>
<td>DUMP</td>
</tr>
<tr>
<td>EXP</td>
<td>FLOOR</td>
</tr>
<tr>
<td>GREATEST</td>
<td>HEXOTRAW</td>
</tr>
<tr>
<td>INITCAP</td>
<td>INSTR</td>
</tr>
<tr>
<td>INSTRB</td>
<td>LEAST</td>
</tr>
<tr>
<td>LENGTH</td>
<td>LENGTHB</td>
</tr>
<tr>
<td>LN</td>
<td>LOG</td>
</tr>
<tr>
<td>LOWER</td>
<td>LPAD</td>
</tr>
<tr>
<td>LTRIM</td>
<td>MOD</td>
</tr>
<tr>
<td>NLS_INITCAP</td>
<td>NLS_UPPER</td>
</tr>
<tr>
<td>NLS_LOWER</td>
<td>NLSSORT</td>
</tr>
<tr>
<td>NVL2</td>
<td>POWER</td>
</tr>
<tr>
<td>RAWTOHEX</td>
<td>REPLACE</td>
</tr>
<tr>
<td>REVERSE</td>
<td>ROUND</td>
</tr>
<tr>
<td>RPAD</td>
<td>RTRIM</td>
</tr>
<tr>
<td>SIGN</td>
<td>SIN</td>
</tr>
<tr>
<td>SINH</td>
<td>SQRT</td>
</tr>
<tr>
<td>STDDEV</td>
<td>SUBSTR</td>
</tr>
</tbody>
</table>
4.6.2 SQL Functions That Can Be Disabled

The following SQL functions are enabled (on) by default:

- GROUPBY
- HAVING
- ORDERBY
- WHERE

ORDERBY controls sort order, which may differ at various sort locations. For example, with ORDERBY ON, a DB2 sort would be based on Extended Binary Coded Decimal Interchange Code (EBCDIC) sorting order, whereas with ORDERBY OFF, an Oracle database sort would be based on ASCII sorting order.

The other three functions, GROUPBY, HAVING, and WHERE, can take additional processing time. If you need to minimize the use of expensive resources, then you should choose the settings of these functions so that the processing is performed with cheaper resource. The above listed functions can also be disabled.

4.6.3 SQL Set Operators and Clauses

The WHERE and HAVING clauses are compatible for all versions of the DRDA server. This means that these clauses are passed unchanged to the DRDA server for processing. Whether clauses GROUP BY and ORDER BY are passed to the DRDA server, or compensated by Oracle database, is determined by the Native Semantics Parameters (see the previous section).

The set operators UNION and UNION ALL are compatible for all versions of the DRDA server, meaning that they are passed unchanged to the DRDA server for processing. The set operators INTERSECT and MINUS are compensated on all versions of the DRDA server except DB2/UBD. For DB2/UBD, INTERSECT is compatible and MINUS is translated to EXCEPT.

4.7 DRDA Data type to Oracle Data type Conversion

To move data between applications and the database, the gateway binds data values from a host variable or literal of a specific data type to a data type understood by the database. Therefore, the gateway maps values from any version of the DRDA server into appropriate Oracle data types before passing these values back to the application or Oracle tool.
Table 4-4 lists the data type mapping and restrictions. The DRDA server data types that are listed in the table are general. Refer to documentation for your DRDA database for restrictions on data type size and value limitations.

### Table 4-4 Data Type Mapping and Restrictions

<table>
<thead>
<tr>
<th>DRDA server</th>
<th>Oracle External</th>
<th>Criteria</th>
<th>If Oracle uses large varchar (32k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR (N)</td>
<td>CHAR (N)</td>
<td>N &lt;= 2000</td>
<td></td>
</tr>
<tr>
<td>VARCHAR (N)</td>
<td>VARCHAR2 (N)</td>
<td>N &lt;= 4000</td>
<td>N &lt;= 32767</td>
</tr>
<tr>
<td></td>
<td>LONG</td>
<td>4000 &lt; N</td>
<td>32767 &lt; N</td>
</tr>
<tr>
<td>LONG VARCHAR (N)</td>
<td>VARCHAR2 (N)</td>
<td>N &lt;= 4000</td>
<td>N &lt;= 32767</td>
</tr>
<tr>
<td>LONG VARCHAR (N)</td>
<td>LONG</td>
<td>4000 &lt; N</td>
<td>32767 &lt; N</td>
</tr>
<tr>
<td>CHAR (N) FOR BIT DATA</td>
<td>RAW (N)</td>
<td>N &lt;= 255</td>
<td></td>
</tr>
<tr>
<td>VARCHAR (N) FOR BIT DATA</td>
<td>RAW (N)</td>
<td>1 &lt;= N &lt;= 2000</td>
<td>1 &lt;= N &lt;= 32767</td>
</tr>
<tr>
<td>VARCHAR (N) FOR BIT DATA</td>
<td>LONG RAW</td>
<td>2000 &lt; N</td>
<td>N &lt; 32767</td>
</tr>
<tr>
<td>LONG VARCHAR (N) FOR BIT DATA</td>
<td>RAW (N)</td>
<td>1 &lt;= N &lt;= 2000</td>
<td>1 &lt;= N &lt;= 32767</td>
</tr>
<tr>
<td></td>
<td>LONG RAW</td>
<td>2000 &lt; N</td>
<td>N &lt; 32767</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td>Refer to performing Date and Time Operations</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>CHAR(8)</td>
<td>See Performing Date and Time Operations</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>CHAR(26)</td>
<td>See Performing Date and Time Operations</td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>CHAR(2N)</td>
<td>N &lt;= 1000</td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td>VARCHAR2 (2N)</td>
<td>N &lt;= 2000</td>
<td>N &lt;= 16370</td>
</tr>
<tr>
<td></td>
<td>LONG</td>
<td>2000 &lt;= N</td>
<td>16370 &lt; N</td>
</tr>
<tr>
<td>LONG VARGRAPHIC</td>
<td>VARCHAR2 (2N)</td>
<td>N &lt;= 2000</td>
<td>N &lt;= 16370</td>
</tr>
<tr>
<td></td>
<td>LONG</td>
<td>2000 &lt; N</td>
<td>16370 &lt; N</td>
</tr>
<tr>
<td>Floating Point Single</td>
<td>FLOAT(24)</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>Floating Point Double</td>
<td>FLOAT(53)</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>Decimal (P, S)</td>
<td>NUMBER (P,S)</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td>LONG</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td>LONG RAW (N)</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>DBLOB</td>
<td>LONG</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td>NUMBER (5)</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>INTEGER</td>
<td>NUMBER (10)</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>

### 4.7.1 Performing Character String Operations

The gateway performs all character string comparisons, concatenations, and sorts using the data type of the referenced columns, and determines the validity of character string values passed by applications using the gateway. The gateway automatically
converts character strings from one data type to another and converts between character strings and dates when needed.

Frequently, DRDA databases are designed to hold non-character binary data in character columns. Applications executed on DRDA systems can generally store and retrieve data as though it contained character data. However, when an application accessing this data runs in an environment that uses a different character set, inaccurate data may be returned.

With the gateway running on the host, character data retrieved from a DB2 UDB for iSeries or DB2 UDB for z/OS host is translated from EBCDIC to ASCII. When character data is sent to DB2 UDB for iSeries or DB2 UDB for z/OS from the host, ASCII data is translated to EBCDIC. When the characters are binary data in a character column, this translation causes the application to receive incorrect information or errors. To resolve these errors, character columns on DB2 UDB for iSeries or DB2 UDB for z/OS that hold non-character data must be created with the FOR BIT DATA option. In the application, the character columns holding non-character data should be processed using the Oracle data types RAW and LONG RAW. The DESCRIBE information for a character column defined with FOR BIT DATA on the host always indicates RAW or LONG RAW.

4.7.2 Converting Character String Data types

The gateway binds character string data values from host variables as fixed-length character strings. The bind length is the length of the character string data value. The gateway performs this conversion on every bind.

The DRDA VARCHAR data type can be between 1 and 32767 characters in length if the Oracle database is configured to use maximum VARCHAR2 size of 32767. Otherwise, the limit is 4000. If the DRDA VARCHAR data type is greater than the Oracle configured VARCHAR2 limit size, then it is converted to an Oracle LONG data type.

The DB2 VARCHAR data type can be no longer than 32767 bytes, which is much shorter than the maximum size for the Oracle LONG data type. If you define an Oracle LONG data type larger than 32767 bytes in length, then you receive an error message when it is mapped to the DB2 VARCHAR data type.

4.7.3 Performing Graphic String Operations

DB2 GRAPHIC data types store only double-byte string data. Sizes for DB2 GRAPHIC data types typically have maximum sizes that are half that of their character counterparts. For example, the maximum size of a CHAR may be 255 characters, whereas the maximum size of a GRAPHIC may be 127 characters.

Oracle database does not have a direct matching data type, and the gateway therefore converts between Oracle character data types to DB2 Graphic data types. Oracle database character data types may contain single, mixed, or double-byte character data. The gateway converts the string data into appropriate double-byte-only format depending upon whether the target DB2 column is a Graphic type and whether Gateway Initialization parameters are set to perform this conversion. For more configuration information, refer to Initialization Parameters.
4.7.4 Performing Date and Time Operations

The implementation of date and time data differs significantly in IBM DRDA databases and Oracle database. Oracle database has a single date data type, **DATE**, which can contain both calendar date and time of day information.

IBM DRDA databases support the following three distinct date and time data types:

- **DATE** is the calendar date only.
- **TIME** is the time of day only.
- **TIMESTAMP** is a numerical value combining calendar date and time of day with microsecond resolution in the internal format of the IBM DRDA database.

4.7.4.1 Processing **TIME** and **TIMESTAMP** Data

There is no built-in mechanism that translates the IBM **TIME** and **TIMESTAMP** data to Oracle **DATE** data. An application must process **TIME** data types to the Oracle **CHAR** format with a length of eight bytes. An application must process the **TIMESTAMP** data type in the Oracle **CHAR** format with a length of 26 bytes.

An application reads **TIME** and **TIMESTAMP** functions as character strings and converts or subsets portions of the string to perform numerical operations. **TIME** and **TIMESTAMP** values can be sent to a DRDA server as character literals or bind variables of the appropriate length and format.

4.7.4.2 Processing **DATE** Data

Oracle and IBM **DATE** data types are mapped to each other. If an IBM **DATE** is queried, then it is converted to an Oracle **DATE** with a zero (midnight) time of day. If an Oracle **DATE** is processed against an IBM **DATE** column, then the date value is converted to the IBM **DATE** format, and any time value is discarded.

Character representations of dates are different in Oracle format and IBM DRDA format. When an Oracle application SQL statement contains a date literal, or conveys a date using a character bind variable, the gateway must convert the date to an IBM DRDA compatible format.

The gateway does not automatically recognize when a character value is being processed against an IBM **DATE** column. Applications are required to distinguish character date values by enclosing them with Oracle **TO_DATE** function notation. For example, if EMP is a synonym or view that accesses data on an IBM DRDA database, then you should not use the following SQL statement:

```
SELECT * FROM EMP WHERE HIREDATE = '03-MAR-81'
```

you should use the following:

```
SELECT * FROM EMP WHERE HIREDATE = TO_DATE('03-MAR-81')
```

In a programmatic interface program that uses a character bind variable for the qualifying date value, you must use this SQL statement:

```
SELECT * FROM EMP WHERE HIREDATE = TO_DATE(:1)
```
The above SQL notation does not affect SQL statement semantics when the statement is executed against an Oracle database table. The statement remains portable across Oracle and IBM DRDA-accessed data stores.

Any date literal other than insert value is checked to match the Oracle `NLS_DATE_FORMAT` before sending to DB2 for processing. TG4DB2 v10.2 does not check to match the `NLS_DATE_FORMAT` format. If such compatibility is desired, then you need to specify `NODATECHK/ON` value as part of `HS_FDS_CAPABILITY` parameter. You can then use any DB2 acceptable date formats:

- YYYY-MM-DD (ISO/JIS)
- DD.MM.YYYY (European)
- MM/DD/YYYY (USA)

For example:

```
SELECT * FROM EMP WHERE HIREDATE = '1981-03-03'
```

The `TO_DATE` requirement also does not pertain to input bind variables that are in Oracle date 7-byte binary format. The gateway recognizes such values as dates. For DB2 UDB for z/OS, if you install the gateway supplied `DATE EXIT`, then you can also use two additional Oracle date formats: DD-MON-RR and DD-MON-YYYY

### 4.7.4.3 Performing Date Arithmetic

The following forms of SQL expression generally do not work correctly with the gateway:

- `date + number`
- `number + date`
- `date - number`
- `date1 - date2`

The date and number addition and subtraction (`date + number, number + date, date - number`) forms are sent through to the DRDA server, where they are rejected. The supported servers do not permit number addition or subtraction with dates.

Because of differing interpretations of date subtraction in the supported servers, subtracting two dates (`date1 - date2`) gives results that vary by server.

> **Note:**

Avoid date arithmetic expressions in all gateway SQL until date arithmetic problems are resolved.

### 4.7.5 Dates

Date handling has two categories:

- Two-digit year dates, which are treated as occurring 50 years before or 50 years after the year 2000.
- Four-digit year dates, which are not ambiguous with regard to the year 2000.

Use one of the following methods to enter twenty-first century dates:
• The **TO_DATE** function

Use any date format including a four-character year field. Refer to the *Oracle Database SQL Language Reference* for the available date format string options.

For example, **TO_DATE('2008-07-23', 'YYYY-MM-DD')** can be used in any **SELECT**, **INSERT**, **UPDATE**, or **DELETE** statement.

• The **NLS_DATE_FORMAT** parameter

**ALTER SESSION SET NLS_DATE_FORMAT** should be used to set the date format used in SQL.

### 4.7.6 NLS_DATE_FORMAT Support

The following table lists the four patterns that can be used for the **NLS_DATE_FORMAT** in **ALTER SESSION SET NLS_DATE_FORMAT**:

<table>
<thead>
<tr>
<th>DB2 Date Format</th>
<th>Pattern</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>EUR</td>
<td>DD.MM.YYYY</td>
<td>30.10.1994</td>
</tr>
<tr>
<td>ISO</td>
<td>YYYY-MM-DD</td>
<td>1994-10-30</td>
</tr>
<tr>
<td>JIS</td>
<td>YYYY-MM-DD</td>
<td>1994-10-30</td>
</tr>
<tr>
<td>USA</td>
<td>MM/DD/YYYY</td>
<td>10/30/1994</td>
</tr>
</tbody>
</table>

The Oracle database default format of ‘DD-MON-YY’ is not permitted with DB2.

The following example demonstrates how to enter and select date values in the twenty-first century:

```sql
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD';
INSERT INTO EMP (HIREDATE) VALUES ('2008-07-23');
SELECT * FROM EMP WHERE HIREDATE = '2008-07-23';
UPDATE EMP SET HIREDATE = '2008-07-24'
    WHERE HIREDATE = '2008-07-23';
DELETE FROM EMP WHERE HIREDATE = '2008-07-24';
```

### 4.7.7 Oracle **TO_DATE** Function

The Oracle **TO_DATE** function is preprocessed in SQL **INSERT**, **UPDATE**, **DELETE**, and **SELECT** **WHERE** clauses. **TO_DATE** functions in **SELECT** result lists are not preprocessed.

The **TO_DATE** function is often needed to provide values to update or compare with date columns. Therefore, the gateway replaces the information included in the **TO_DATE** clause with an acceptable value before the SQL statement is sent to DB2.

Except for the **SELECT** result list, all **TO_DATE** functions are preprocessed and turned into values that are the result of the **TO_DATE** function. Only **TO_DATE(literal)** or **TO_DATE(:bind_variable) is permitted.** Except in **SELECT** result lists, the **TO_DATE(column_name)** function format is not supported.

The preprocessing of the Oracle **TO_DATE** functions into simple values is useful in an **INSERT VALUES** clause because DB2 does not allow functions in the **VALUES** clause. In this case, DB2 receives a simple value in the **VALUES** list. All forms of the **TO_DATE** function (with one, two, or three operands) are supported.
4.7.8 Performing Numeric data type Operations

IBM versions of the DRDA server perform automatic conversions to the numeric data type of the destination column (such as integer, double-precision floating point, or decimal). The user has no control over the data type conversion, and this conversion can be independent of the data type of the destination column in the database.

For example, if PRICE is an integer column of the PRODUCT table in an IBM DRDA database, then the update shown in the following example inaccurately sets the price of an ice cream cone to $1.00 because the IBM DRDA server automatically converts a floating point to an integer:

```
UPDATE PRODUCT
SET PRICE = 1.50
WHERE PRODUCT_NAME = 'ICE CREAM CONE';
```

Because PRICE is an integer, the IBM DRDA server automatically converts the decimal data value of 1.50 to 1.

4.7.9 Mapping the COUNT Function

Oracle database supports the following four operands for the COUNT function:

- COUNT(*)
- COUNT(DISTINCT colname)
- COUNT(ALL colname)
- COUNT(colname)

Some DRDA servers do not support all forms of COUNT, specifically COUNT(colname) and COUNT(ALL colname). In those cases the COUNT function and its arguments are translated into COUNT(*). This may not yield the desired results, especially if the column being counted contains NULLs.

For those DRDA servers that do not support the above forms, it may be possible to achieve equivalent functionality by adding a WHERE clause. For example,

```
SELECT COUNT(colname) FROM table@dblink WHERE colname IS NOT NULL
```

or

```
SELECT COUNT(ALL colname) FROM table@dblink WHERE colname IS NOT NULL
```

You can also use native semantics to indicate support for all COUNT functions with the following parameter in your gateway initialization file:

```
HS_FDS_CAPABILITY=(COUNTCOL=YES)
```

Refer to SQL Limitations for known DRDA servers that do not support all forms of COUNT.

4.7.10 Performing Zoned Decimal Operations

A zoned decimal field is described as packed decimal on Oracle database. However, an Oracle application such as a Pro*C program can insert into a zoned decimal column using any supported Oracle numeric data type. The gateway converts this...
number into the most suitable data type. Data can be fetched from a DRDA database into any Oracle data type, provided that it does not result in a loss of information.

4.8 Passing Native SQL Statements through the Gateway

The passthrough SQL feature enables an application developer to send a SQL statement directly to the DRDA server without the statement being interpreted by Oracle database. `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` SQL passthrough statements that are supported by the gateway are limited to nonqueries (INSERT, UPDATE, DELETE, and DDL statements) and cannot contain bind variables. The gateway can run native SQL statements using `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE`.

`DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` is a built-in gateway function. This function receives one input argument and returns the number of rows affected by the SQL statement. For data definition language (DDL) statements, the function returns zero.

`DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` are reserved names of the gateway and are used specifically for running native SQL.

The 12c Release 2 (12.2) of Oracle Database Gateway for DRDA enables retrieval of result sets from queries issued with passthrough. The syntax is different from the `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` function. Refer to "Retrieving Results Sets Through Passthrough" for more information.

4.8.1 Processing DDL Statements through Passthrough

As noted above, SQL statements that are processed through the `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` function are not interpreted by the Oracle database. As a result, the Oracle database will not know if such statements are making any modifications to the DRDA server. This means that unless you keep the cached information of the Oracle database up to date after changes to the DRDA server, the database may continue to rely upon inaccurate or outdated information in subsequent queries within the same session.

An example of this occurs when you alter the structure of a table by either adding or removing a column. When an application references a table through the gateway (for example, when you perform a query on it), the Oracle database caches the table definition. Now, suppose that within the same session, the application subsequently alters the table's form, by using `DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE` to add a column. Then, the next reference to the table by the application will return the old column definitions of the table and will ignore the table's new column. This is because the Oracle database did not process the statement and, so, has no knowledge of the alteration. Because the database does not know of the alteration, it has no reason to requery the table form, and, so, it will use the already-cached form to handle any new queries.

In order for the Oracle database to acquire the new form of the table, the existing session with the gateway must be closed and a new session must be opened. This can be accomplished in either of two ways:

- By ending the application session with the Oracle database and starting a new session after modifications have been made to the DRDA server; or
- By running the `ALTER SESSION CLOSE DATABASE LINK` command after making any modifications to the DRDA server.
Either of the above actions will void the cached table definitions and will force the Oracle database to acquire new definitions on the next reference.

4.8.2 Using DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE

To run a passthrough SQL statement using DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE, use the following syntax:

\[
\text{number_of_rows} = \text{DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@dblink} ('\text{native_DRDA_sql}')
\]

where:

- \text{number_of_rows} is a variable that is assigned the number of rows affected by the passthrough SQL completion. For DDL statements, a zero is returned for the number of rows affected.
- \text{dblink} is the name of the database link used to access the gateway.
- \text{native_DRDA_sql} is a valid nonquery SQL statement (except \text{CONNECT}, \text{COMMIT}, and \text{ROLLBACK}). The statement cannot contain bind variables. The DRDA server rejects native SQL statements that cannot be dynamically prepared. The SQL statement passed by the DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE function must be a character string. For more information regarding valid SQL statements, refer to the SQL Reference for the particular DRDA server.

4.8.2.1 Examples

1. Insert a row into a DB2 table using DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE:

   ```sql
   DECLARE
       num_rows integer;
   BEGIN
       num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@dblink ('INSERT INTO SCOTT.DEPT VALUES (10,'"PURCHASING","PHOENIX")');
   END;
   /
   ```

2. Create a table in DB2 using DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE:

   ```sql
   DECLARE
       num_rows integer;
   BEGIN
       num_rows:=DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE@dblink ('CREATE TABLE MYTABLE (COL1 INTEGER, COL2 INTEGER, COL3 CHAR(14), COL4 VARCHAR(13))');
   END;
   /
   ```

4.8.3 Retrieving Results Sets Through Passthrough

Oracle Database Gateway for DRDA provides a facility to retrieve results sets from a SELECT SQL statement entered through passthrough. Refer to Oracle Database Heterogeneous Connectivity User's Guide for additional information.

4.8.3.1 Example

```sql
DECLARE
    CRS binary_integer;
```
4.9 Oracle Data Dictionary Emulation on a DRDA Server

The gateway optionally augments the DRDA database catalogs with data dictionary views modeled on the Oracle data dictionary. These views are based on the dictionary tables in the DRDA database, presenting that catalog information in views familiar to Oracle users. The views created during the installation of the gateway automatically limit the data dictionary information presented to each user based on the privileges of that user.

4.9.1 Using the Gateway Data Dictionary

The gateway data dictionary views provide users with an Oracle-like interface to the contents and use of the DRDA database. Oracle products require some of these views. The gateway supports the DB2 UDB for z/OS, DB2 UDB for iSeries, and DB2/UDB catalog views.

You can query the gateway data dictionary views to see the objects in the DRDA database and to determine the authorized users of the DRDA database. The Oracle Database Gateway for DRDA supports many Oracle catalog views. Refer to Oracle DB2 Data Dictionary Views for descriptions of Oracle DB2 catalog views. These views are completely compatible with the gateway.

4.9.2 Using the DRDA Catalog

Each DRDA database has its own catalog tables and views, which you might find useful. Refer to the appropriate IBM documentation for descriptions of these catalogs.
5

Error Messages, Diagnosis, and Reporting

The following sections provide information about error messages and error codes specific to the Oracle Database Gateway for DRDA:

- Interpreting Gateway Error Messages
- Mapped Errors
- SQL Tracing and the Gateway

5.1 Interpreting Gateway Error Messages

The gateway architecture consists of different components. Any component may detect and report an error condition while processing SQL statements that refer to one or more DRDA database tables. This means that errors can be complex, involving error codes and supporting data from multiple components. In all cases, however, the application ultimately receives a single error code or a return code.

As most gateway messages exceed the 70 character message area in the Oracle SQL Communications Area (SQLCA), the programmatic interfaces and Oracle Call Interfaces, that you use to access data through the gateway should use SQLGLM or OERHMS to view the entire text of messages. Refer to the programmer's guide to the Oracle precompilers for additional information about SQLGLM, and refer to the Oracle C++ Call Interface Programmer's Guide for additional information about OERHMS.

Errors encountered when using the gateway can originate from many sources, as follows:

- Errors detected by the Oracle database
- Errors detected by the gateway
- Errors detected in the DRDA software, either on the client or server side
- Communication errors
- Errors detected by the server database

5.1.1 Errors Detected by the Gateway

Errors detected by the Oracle database are reported back to the application or tool with the standard ORA type message. Refer to Oracle Database Error Messages for descriptions of these errors. For example, the following error occurs when an undefined database link name is specified:

ORA-02019: connection description for remote database not found

Errors in the ORA-9100 to ORA-9199 range are reserved for the generic gateway layer (components of the gateway that are not specific to DRDA). Messages in this range are documented in Oracle Database Error Messages.
5.1.2 Errors Detected in the DRDA Software

Errors detected in the DRDA gateway, on the client or server side, are usually reported with error ORA-28500, followed by a gateway-specific expanded error message. There are two return codes reported in the expanded message:

- **drc** specifies DRDA-specific errors.
- **grc** specifies generic gateway errors detected in the DRDA layer. These errors are documented in the *Oracle Database Error Messages*.

**Note:**

Error code ORA-28500 was error code ORA-09100 prior to gateway version 8. Error code ORA-28501 was listed as ORA-09101 prior to gateway version 8.

The values in parentheses that follow the drc values are used for debugging by Oracle Support Services. The errp field indicates the program (client or server) that detected the error. If present, errmc lists any error tokens.

For example, the following error message is returned when the database name specified with the **DRDA_REMOTE_NAME** parameter in the *initsid.ora* file is not defined at the DRDA server:

```
ORA-28500: connection from ORACLE to non-Oracle system returned the message:
```

5.1.3 Errors Detected by the DRDA Server

Errors detected by the DRDA server are reported with an ORA-28500 followed by a gateway-specific expanded error message. Refer to IBM documentation for the specific database being used. Also refer to Mapped Errors for some SQL errors that get translated.

**Note:**

Error code ORA-28500 was error code ORA-09100 prior to gateway version 8. Error code ORA-28501 was listed as ORA-09101 prior to gateway version 8.

For example, the following error message indicates that the DRDA server did not find the DB2 database name specified in the **HS_FDS_CONNECT_INFO** parameter in the *initSID.ora* file:

```
ORA-28500: connection from ORACLE to a non-Oracle system returned this message:
Oracle [ODBC DB2 Wire Protocol driver] Remote Database Not Found: UNKNOWN
```
5.2 Mapped Errors

Some SQL errors are returned from the DRDA server and are translated to an Oracle error code. This is needed when the Oracle instance or gateway provides special handling of an error condition.

The following is an example of a translated object does not exist error:

ORA-00942: table or view does not exist
[Oracle][ODBC DB2 Wire Protocol driver][UDB DB2 for OS/390 and z/OS]PCASTRO.XXX IS AN UNDEFINED NAME.

5.3 SQL Tracing and the Gateway

When developing applications, it is often useful to be able to see the exact SQL statements that are being passed through the gateway. The following sections describe setting appropriate trace parameters and setting up the debug gateway.

5.3.1 SQL Tracing in the Oracle Database

Oracle database has a command for capturing the SQL statement that is actually sent to the gateway. This command is called EXPLAIN PLAN. The EXPLAIN PLAN command is used to determine the execution plan that Oracle database follows to execute a specified SQL statement. This command inserts a row, which describes each step of the execution plan, into a specified table. If you are using cost-based optimization, then this command also determines the cost of executing the statement. The syntax of the command is:

```
EXPLAIN PLAN [ SET STATEMENT_ID = 'text' ]
             [ INTO [schema.]table[@dblink] ] FOR statement
```

For detailed information on this command, refer to the Oracle Database SQL Language Reference.

---

**Note:**

In most cases, EXPLAIN PLAN should be sufficient to extract the SQL statement that is actually sent to the gateway, and thus sent to the DRDA server. However, certain SQL statement form have post-processing performed on them in the gateway.
Oracle DB2 Data Dictionary Views

The following section covers the Oracle Database Gateway for DRDA data dictionary views accessible to all users of Oracle database. Any user with SELECT privileges for DB2 catalog tables can access most of the views.

N/A is used in the tables to denote that the column is not valid for the gateway.

A.1 Supported Views

The following is a list of Oracle data dictionary views that are supported by the gateway for DB2 UDB for z/OS, DB2 UDB for iSeries, and DB2/UDB DRDA servers.

- ALL_CATALOG
- ALL_COL_COMMENTS
- ALL_CONS_COLUMNS
- ALL_CONSTRAINTS
- ALL_INDEXES
- ALL_IND_COLUMNS
- ALL_OBJECTS
- ALL_SYNONYMS
- ALL_TAB_COMMENTS
- ALL_TABLES
- ALL_TAB_COLUMNS
- ALL_USERS
- ALL_VIEWS
- COL_PRIVILEGES
- DICTIONARY
- DUAL
- TABLE_PRIVILEGES
- USER_CATALOG
- USER_COL_COMMENTS
- USER_CONSTRAINTS
- USER_CONS_COLUMNS
- USER_INDEXES
- USER_OBJECTS
- USER_SYNONYMS
A.2 ALL_CATALOG

The ALL_CATALOG view contains all tables, views, synonyms, and sequence accessible to the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the object</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the object</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>Type of object</td>
</tr>
</tbody>
</table>

A.3 ALL_COL_COMMENTS

The ALL_COL_COMMENTS view contains comments on columns of accessible tables and views.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the object</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Object name</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>Column name</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>Comments on column</td>
</tr>
</tbody>
</table>

A.4 ALL_CONS_COLUMNS

The ALL_CONS_COLUMNS view contains information about accessible columns in constraint definitions.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the constraint definition</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td>Name of the constraint definition</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the table with a constraint definition</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>Name of the column specified in the constraint definition</td>
</tr>
<tr>
<td>POSITION</td>
<td>Original position of column in definition</td>
</tr>
</tbody>
</table>

A.5 ALL_CONSTRAINTS

The ALL_CONSTRAINTS view contains constraint definitions on accessible tables.
<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the constraint definition</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td>Name of the constraint definition</td>
</tr>
<tr>
<td>CONSTRAINT_TYPE</td>
<td>Type of the constraint definition</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the table with constraint definition</td>
</tr>
<tr>
<td>SEARCH_CONDITION</td>
<td>Text of the search condition for table check</td>
</tr>
<tr>
<td>R_OWNER</td>
<td>Owner of the table used in referential constraint</td>
</tr>
<tr>
<td>R_CONSTRAINT_NAME</td>
<td>Name of the unique constraint definition for referenced table</td>
</tr>
<tr>
<td>DELETE_RULE</td>
<td>Delete rule for a referential constraint</td>
</tr>
<tr>
<td>STATUS</td>
<td>Status of a constraint</td>
</tr>
<tr>
<td>DEFFERRABLE</td>
<td>Whether the constraint is deferrable</td>
</tr>
<tr>
<td>DEFERRED</td>
<td>Whether the constraint was initially deferred</td>
</tr>
<tr>
<td>VALIDATED</td>
<td>Whether all data obeys the constraint</td>
</tr>
<tr>
<td>GENERATED</td>
<td>Whether the name of the constraint is user or system generated</td>
</tr>
<tr>
<td>BAD</td>
<td>Constraint specifies a century in an ambiguous manner</td>
</tr>
<tr>
<td>RELY</td>
<td>Whether an enabled constraint is enforced or unenforced</td>
</tr>
<tr>
<td>LAST_CHANGE</td>
<td>When the constraint was last enabled</td>
</tr>
<tr>
<td>INDEX_OWNER</td>
<td>N/A</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**A.6 ALL_INDEXES**

The **ALL_INDEXES** view contains description of indexes on tables accessible to the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the index</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>Name of the index</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>Type of the index</td>
</tr>
<tr>
<td>TABLE_OWNER</td>
<td>Owner of the indexed object</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the indexed object</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>Type of the indexed object</td>
</tr>
<tr>
<td>UNIQUENESS</td>
<td>Uniqueness status of the index</td>
</tr>
<tr>
<td>COMPRESSION</td>
<td>N/A</td>
</tr>
<tr>
<td>PREFIX_LENGTH</td>
<td>0</td>
</tr>
<tr>
<td>TABLESPACE_NAME</td>
<td>Name of the tablespace containing the index</td>
</tr>
<tr>
<td>INI_TRANS</td>
<td>N/A</td>
</tr>
<tr>
<td>MAX_TRANS</td>
<td>N/A</td>
</tr>
<tr>
<td>Column name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INITIAL_EXTENT</td>
<td>N/A</td>
</tr>
<tr>
<td>NEXT_EXTENT</td>
<td>N/A</td>
</tr>
<tr>
<td>MIN_EXTENTS</td>
<td>N/A</td>
</tr>
<tr>
<td>MAX_EXTENTS</td>
<td>N/A</td>
</tr>
<tr>
<td>PCT_INCREASE</td>
<td>N/A</td>
</tr>
<tr>
<td>PCT_THRESHOLD</td>
<td>Threshold percentage of block space allowed per index entry</td>
</tr>
<tr>
<td>INCLUDE_COLUMN</td>
<td>Column ID of the last column to be included in an index-organized table</td>
</tr>
<tr>
<td>FREELISTS</td>
<td>Number of process freelists allocated to this segment</td>
</tr>
<tr>
<td>FREELIST_GROUPS</td>
<td>Number of freelist groups allocated to this segment</td>
</tr>
<tr>
<td>PCT_FREE</td>
<td>N/A</td>
</tr>
<tr>
<td>LOGGING</td>
<td>Logging information</td>
</tr>
<tr>
<td>BLEVEL</td>
<td>Depth of the index from its root block to its leaf blocks. A depth of 1 indicates that the root block and the leaf block are the same.</td>
</tr>
<tr>
<td>LEAF_BLOCKS</td>
<td>Number of leaf blocks in the index</td>
</tr>
<tr>
<td>DISTINCT_KEYS</td>
<td>Number of distinct indexed values. For indexes that enforce UNIQUE and PRIMARY KEY constraints, this value is the same as the number of rows in the table.</td>
</tr>
<tr>
<td>AVG_LEAF_BLOCKS_PER_KEY</td>
<td>N/A</td>
</tr>
<tr>
<td>AVG_DATA_BLOCKS_PER_KEY</td>
<td>N/A</td>
</tr>
<tr>
<td>CLUSTERING_FACTOR</td>
<td>N/A</td>
</tr>
<tr>
<td>STATUS</td>
<td>State of the index: VALID</td>
</tr>
<tr>
<td>NUM_ROWS</td>
<td>Number of rows in the index</td>
</tr>
<tr>
<td>SAMPLE_SIZE</td>
<td>Size of the sample used to analyze the index</td>
</tr>
<tr>
<td>LAST_ANALYZED</td>
<td>Date on which an index was most recently analyzed</td>
</tr>
<tr>
<td>DEGREE</td>
<td>Number of threads per instance for scanning the index</td>
</tr>
<tr>
<td>INSTANCES</td>
<td>Number of instances across which the index is to be scanned</td>
</tr>
<tr>
<td>PARTITIONED</td>
<td>Whether the index is partitioned</td>
</tr>
<tr>
<td>TEMPORARY</td>
<td>Whether the index is on a temporary table</td>
</tr>
<tr>
<td>GENERATED</td>
<td>Whether the name of the index is system generated</td>
</tr>
<tr>
<td>SECONDARY</td>
<td>N/A</td>
</tr>
<tr>
<td>BUFFER_POOL</td>
<td>Whether the index is a secondary object</td>
</tr>
<tr>
<td>USER_STATS</td>
<td>N/A</td>
</tr>
<tr>
<td>DURATION</td>
<td>N/A</td>
</tr>
<tr>
<td>PCT_DIRECT_ACCESS</td>
<td>N/A</td>
</tr>
<tr>
<td>ITYP_OWNER</td>
<td>N/A</td>
</tr>
<tr>
<td>ITYP_NAME</td>
<td>N/A</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>N/A</td>
</tr>
</tbody>
</table>
The `ALL_IND_COLUMNS` view contains the columns of indexes on all tables that are accessible to the current user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX_OWNER</td>
<td>Owner of the index</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>Name of the index</td>
</tr>
<tr>
<td>TABLE_OWNER</td>
<td>Owner of the table or cluster</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the table or cluster</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>Column name or attribute of object type column</td>
</tr>
<tr>
<td>COLUMN_POSITION</td>
<td>Position of a column or attribute within the index</td>
</tr>
<tr>
<td>COLUMN_LENGTH</td>
<td>Indexed length of the column</td>
</tr>
<tr>
<td>CHAR_LENGTH</td>
<td>Maximum codepoint length of the column</td>
</tr>
<tr>
<td>DESCEND</td>
<td>Whether the column is sorted in descending order (Y/N)</td>
</tr>
</tbody>
</table>

The `ALL_OBJECTS` view contains objects accessible to the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the object</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>Name of object</td>
</tr>
<tr>
<td>SUBOBJECT_NAME</td>
<td>Name of the subobject</td>
</tr>
<tr>
<td>OBJECT_ID</td>
<td>Object number of the object</td>
</tr>
<tr>
<td>DATA_OBJECT_ID</td>
<td>Dictionary object number of the segment that contains the object</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>Type of object</td>
</tr>
<tr>
<td>CREATED</td>
<td>N/A</td>
</tr>
<tr>
<td>LAST_DDL_TIME</td>
<td>N/A</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>N/A</td>
</tr>
<tr>
<td>STATUS</td>
<td>State of the object</td>
</tr>
<tr>
<td>TEMPORARY</td>
<td>Whether the object is temporary</td>
</tr>
</tbody>
</table>

A.7 ALL_IND_COLUMNS

A.8 ALL_OBJECTS
### Appendix A

#### A.9 ALL_SYNONYMS

The **ALL_SYNONYMS** view contains all synonyms accessible to the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERATED</td>
<td>Whether the name of this object system is generated</td>
</tr>
<tr>
<td>SECONDARY</td>
<td>N/A</td>
</tr>
</tbody>
</table>

#### A.10 ALL_TABLES

The **ALL_TABLES** view contains description of tables accessible to the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the table</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the table</td>
</tr>
<tr>
<td>TABLESPACE_NAME</td>
<td>Name of the tablespace containing the table</td>
</tr>
<tr>
<td>CLUSTER_NAME</td>
<td>N/A</td>
</tr>
<tr>
<td>IOT_NAME</td>
<td>Name of the index organized table</td>
</tr>
<tr>
<td>PCT_FREE</td>
<td>N/A</td>
</tr>
<tr>
<td>PCT_USED</td>
<td>N/A</td>
</tr>
<tr>
<td>INIT_TRANS</td>
<td>N/A</td>
</tr>
<tr>
<td>MAX_TRANS</td>
<td>N/A</td>
</tr>
<tr>
<td>INITIAL_EXTENT</td>
<td>N/A</td>
</tr>
<tr>
<td>NEXT_EXTENT</td>
<td>N/A</td>
</tr>
<tr>
<td>MIN_EXTENTS</td>
<td>N/A</td>
</tr>
<tr>
<td>MAX_EXTENTS</td>
<td>N/A</td>
</tr>
<tr>
<td>PCT_INCREASE</td>
<td>N/A</td>
</tr>
<tr>
<td>FREELISTS</td>
<td>Number of process freelists allocated to this segment</td>
</tr>
<tr>
<td>FREELIST_GROUPS</td>
<td>Number of freelist groups allocated to this segment</td>
</tr>
<tr>
<td>LOGGING</td>
<td>Logging attribute</td>
</tr>
<tr>
<td>BACKED_UP</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Column name | Description
--- | ---
NUM_ROWS | Number of rows in the table
BLOCKS | N/A
EMPTY_BLOCKS | N/A
AVG_SPACE | N/A
CHAIN_CNT | N/A
AVG_ROW_LEN | Average length of a row in the table in bytes
AVG_SPACE_FREELIST_BLOCKS | Average freespace of all blocks on a freelist
NUM_FREELIST_BLOCKS | Number of blocks on the freelist
DEGREE | Number of threads per instance for scanning the table
INSTANCES | Number of instances across which the table is to be scanned
CACHE | Whether the cluster is to be cached in the buffer cache
TABLE_LOCK | Whether the table locking is enabled or disabled
SAMPLE_SIZE | Sample size used in analyzing this table
LAST_ANALYZED | Date on which this table was most recently analyzed
PARTITIONED | Whether this table is partitioned
IOT_TYPE | Whether the table is an index-organized table
TEMPORARY | Can the current session only see data that it placed in this object itself?
SECONDARY | N/A
NESTED | Whether the table is a nested table
BUFFER_POOL | Default buffer pool for the object
ROW_MOVEMENT | N/A
GLOBAL_STATS | N/A
USER_STATS | N/A
DURATION | N/A
SKIP_CORRUPT | N/A
MONITORING | N/A
CLUSTER_OWNER | N/A
DEPENDENCIES | N/A
COMPRESSION | N/A

### A.11 ALL_TAB_COLUMNS

The ALL_TAB_COLUMNS view contains columns of all tables, views, and clusters accessible to the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the table or view</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Table or view name</td>
</tr>
</tbody>
</table>
### Column name | Description
--- | ---
COLUMN_NAME | Column name
DATA_TYPE | Data type of the column
DATA_TYPE_MOD | Data type modifier of the column
DATA_TYPE_OWNER | Owner of the data type of the column
DATA_LENGTH | Maximum length of the column in bytes
DATA_PRECISION | N/A
DATA_SCALE | Digits to the right of decimal point in a number
NULLABLE | Whether the column permits nulls? Value is n if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY key.
COLUMN_ID | Sequence number of the column as created
DEFAULT_LENGTH | N/A
DATA_DEFAULT | N/A
NUM_DISTINCT | Number of distinct values in each column of the table
LOW_VALUE | For tables with more than three rows, the second lowest and second highest values. These statistics are expressed in hexadecimal notation for the internal representation of the first 32 bytes of the values.
HIGH_VALUE | N/A
DENSITY | N/A
NUM NULLS | Number of nulls in the column
NUM_BUCKETS | Number of buckets in histogram for the column
LAST_ANALYZED | Date on which this column was most recently analyzed
SAMPLE_SIZE | Sample size used in analyzing this column
CHARACTER_SET_NAME | Name of the character set
CHAR_COL_DECL_LENGTH | Length of the character set
GLOBAL_STATS | N/A
USER_STATS | N/A
AVG_COL_LEN | Average length of the column (in bytes)
CHAR_LENGTH | Displays the length of the column in characters
CHAR_USED | N/A

### A.12 ALL_TAB_COMMENTS

The ALL_TAB_COMMENTS view contains comments on tables and views accessible to the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the object</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the object</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>Type of the object</td>
</tr>
</tbody>
</table>
### A.13 ALL_USERS

The **ALL_USERS** contains information about all users of the database.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERNAME</td>
<td>Name of the user</td>
</tr>
<tr>
<td>USER_ID</td>
<td>N/A</td>
</tr>
<tr>
<td>CREATED</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### A.14 ALL_VIEWS

The **ALL_VIEWS** view contains text of views accessible to the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the view</td>
</tr>
<tr>
<td>VIEW_NAME</td>
<td>Name of the view</td>
</tr>
<tr>
<td>TEXT_LENGTH</td>
<td><strong>ALL_VIEWS</strong> view will return 0 for <strong>TEXT_LENGTH</strong> column</td>
</tr>
<tr>
<td>TEXT</td>
<td><strong>ALL_VIEWS</strong> view will return <strong>NULL</strong> for <strong>TEXT</strong> column</td>
</tr>
<tr>
<td>TYPE_TEXT_LENGTH</td>
<td>Length of the type clause of the typed view</td>
</tr>
<tr>
<td>TYPE_TEXT</td>
<td>Type clause of the typed view</td>
</tr>
<tr>
<td>OID_TEXT_LENGTH</td>
<td>Length of the <strong>WITH OID</strong> clause of the typed view</td>
</tr>
<tr>
<td>OID_TEXT</td>
<td><strong>WITH OID</strong> clause of the typed view</td>
</tr>
<tr>
<td>VIEW_TYPE_OWNER</td>
<td>Owner of the type of the view, if the view is a typed view</td>
</tr>
<tr>
<td>VIEW_TYPE</td>
<td>Type of the view, if the view is a typed view</td>
</tr>
<tr>
<td>SUPERVIEW_NAME</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### A.15 COLUMN_PRIVILEGES

The **COLUMN_PRIVILEGES** view contains grants on columns for which the user is the grantor, grantee, or owner, or, the grantee is **PUBLIC**.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANTEE</td>
<td>Name of the user to whom access was granted</td>
</tr>
<tr>
<td>OWNER</td>
<td>Username of the owner of the object</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the object</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>Name of the column</td>
</tr>
<tr>
<td>GRANTOR</td>
<td>Name of the user who performed the grant</td>
</tr>
<tr>
<td>INSERT_PRIV</td>
<td>Permission to insert into the column</td>
</tr>
</tbody>
</table>
### A.16 DICTIONARY

The DICTIONARY view contains list or data dictionary tables.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_NAME</td>
<td>Table name</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>Description of the table</td>
</tr>
</tbody>
</table>

### A.17 DUAL

The DUAL view contains list of dual tables.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUMMY</td>
<td>A dummy column</td>
</tr>
</tbody>
</table>

### A.18 TABLE_PRIVILEGES

The TABLE_PRIVILEGES view contains grants on objects for which the user is the grantor, grantee, or owner, or, the grantee is PUBLIC.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANTEE</td>
<td>Name of the user to whom access is granted</td>
</tr>
<tr>
<td>OWNER</td>
<td>Owner of the object</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the object</td>
</tr>
<tr>
<td>GRANTOR</td>
<td>Name of the user who performed the grant</td>
</tr>
<tr>
<td>SELECT_PRIV</td>
<td>Permission to select data from an object</td>
</tr>
<tr>
<td>INSERT_PRIV</td>
<td>Permission to insert data into an object</td>
</tr>
<tr>
<td>DELETE_PRIV</td>
<td>Permission to delete data from an object</td>
</tr>
<tr>
<td>UPDATE_PRIV</td>
<td>Permission to update an object</td>
</tr>
<tr>
<td>REFERENCES_PRIV</td>
<td>N/A</td>
</tr>
<tr>
<td>ALTER_PRIV</td>
<td>Permission to alter an object</td>
</tr>
<tr>
<td>INDEX_PRIV</td>
<td>Permission to create or drop an index on an object</td>
</tr>
<tr>
<td>CREATED</td>
<td>Timestamp for the grant</td>
</tr>
</tbody>
</table>
A.19 USER_CATALOG

The USER_CATALOG view contains tables, views, synonyms, and sequences owned by the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_NAME</td>
<td>Name of the object</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>Type of the object</td>
</tr>
</tbody>
</table>

A.20 USER_COL_COMMENTS

The USER_COL_COMMENTS view contains comments on columns of tables and views owned by the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_NAME</td>
<td>Name of the object</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>Name of the column</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>Comments on the column</td>
</tr>
</tbody>
</table>

A.21 USER_CONSTRAINTS

The USER_CONSTRAINTS view contains constraint definitions on tables owned by the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the constraint definition</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td>Name associated with the constraint definition</td>
</tr>
<tr>
<td>CONSTRAINT_TYPE</td>
<td>Type of the constraint definition</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name associated with the table with constraint definition</td>
</tr>
<tr>
<td>SEARCH_CONDITION</td>
<td>Text of the search condition for table check</td>
</tr>
<tr>
<td>R_OWNER</td>
<td>Owner of table used in referential constraint</td>
</tr>
<tr>
<td>R_CONSTRAINT_NAME</td>
<td>Name of the unique constraint definition for referenced table</td>
</tr>
<tr>
<td>DELETE_RULE</td>
<td>Delete rule for referential constraint</td>
</tr>
<tr>
<td>STATUS</td>
<td>Status of a constraint</td>
</tr>
<tr>
<td>DEFERRABLE</td>
<td>Whether the constraint is deferrable</td>
</tr>
<tr>
<td>DEFERRED</td>
<td>Whether the constraint was initially deferred</td>
</tr>
<tr>
<td>VALIDATED</td>
<td>Whether all data obeys the constraint</td>
</tr>
<tr>
<td>GENERATED</td>
<td>Whether the name of the constraint is user or system generated</td>
</tr>
<tr>
<td>BAD</td>
<td>Constraint specifies a century in an ambiguous manner</td>
</tr>
<tr>
<td>LAST_CHANGE</td>
<td>When the constraint was last enabled</td>
</tr>
<tr>
<td>INDEX_OWNER</td>
<td>N/A</td>
</tr>
<tr>
<td>INDEX_NAME</td>
<td>N/A</td>
</tr>
</tbody>
</table>
A.22 USER_CONS_COLUMNS

The USER_CONS_COLUMNS contains information about columns in constraint definitions owned by the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>Owner of the constraint definition</td>
</tr>
<tr>
<td>CONSTRAINT_NAME</td>
<td>Name associated with the constraint definition</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name associated with table with constraint definition</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>Name associated with column specified in the constraint definition</td>
</tr>
<tr>
<td>POSITION</td>
<td>Original position of column in definition</td>
</tr>
</tbody>
</table>

A.23 USER_INDEXES

The USER_INDEXES view contains description of the user's indexes:

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX_NAME</td>
<td>Name of the index</td>
</tr>
<tr>
<td>INDEX_TYPE</td>
<td>Type of index</td>
</tr>
<tr>
<td>TABLE_OWNER</td>
<td>Owner of the indexed object</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the indexed object</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>Type of the indexed object</td>
</tr>
<tr>
<td>UNIQUENESS</td>
<td>Uniqueness status of the index</td>
</tr>
<tr>
<td>COMPRESSION</td>
<td>N/A</td>
</tr>
<tr>
<td>PREFIX_LENGTH</td>
<td>0</td>
</tr>
<tr>
<td>TABLESPACE_NAME</td>
<td>Name of the tablespace containing the index</td>
</tr>
<tr>
<td>INI_TRANS</td>
<td>N/A</td>
</tr>
<tr>
<td>MAX_TRANS</td>
<td>N/A</td>
</tr>
<tr>
<td>INITIAL_EXTENT</td>
<td>N/A</td>
</tr>
<tr>
<td>NEXT_EXTENT</td>
<td>N/A</td>
</tr>
<tr>
<td>MIN_EXTENTS</td>
<td>N/A</td>
</tr>
<tr>
<td>MAX_EXTENTS</td>
<td>N/A</td>
</tr>
<tr>
<td>PCT_INCREASE</td>
<td>N/A</td>
</tr>
<tr>
<td>PCT_THRESHOLD</td>
<td>Threshold percentage of block space allowed per index entry</td>
</tr>
<tr>
<td>INCLUDE_COLUMN</td>
<td>Column ID of the last column to be included in index-organized table</td>
</tr>
<tr>
<td>FREELISTS</td>
<td>Number of process freelists allocated to a segment</td>
</tr>
<tr>
<td>FREELIST_GROUPS</td>
<td>Number of freelist groups allocated to a segment</td>
</tr>
<tr>
<td>PCT_FREE</td>
<td>N/A</td>
</tr>
<tr>
<td>LOGGING</td>
<td>Logging information</td>
</tr>
<tr>
<td>Column name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BLEVEL</td>
<td>Depth of the index from its root block to its leaf blocks. A depth of 1 indicates that the root and leaf block are the same.</td>
</tr>
<tr>
<td>LEAF_BLOCKS</td>
<td>Number of leaf blocks in the index</td>
</tr>
<tr>
<td>DISTINCT_KEYS</td>
<td>Number of distinct indexed values. For indexes that enforce <code>UNIQUE</code> and <code>PRIMARY KEY</code> constraints, this value is the same as the number of rows in the table.</td>
</tr>
<tr>
<td>AVG_LEAF_BLOCKS_PER_KEY</td>
<td>N/A</td>
</tr>
<tr>
<td>AVG_DATA_BLOCKS_PER_KEY</td>
<td>N/A</td>
</tr>
<tr>
<td>CLUSTERING_FACTOR</td>
<td>N/A</td>
</tr>
<tr>
<td>STATUS</td>
<td>State of the indexes: <strong>VALID</strong></td>
</tr>
<tr>
<td>NUM_ROWS</td>
<td>Number of rows in the index</td>
</tr>
<tr>
<td>SAMPLE_SIZE</td>
<td>Size of the sample used to analyze the index</td>
</tr>
<tr>
<td>LAST_ANALYZED</td>
<td>Date on which the index was most recently analyzed</td>
</tr>
<tr>
<td>DEGREE</td>
<td>Number of threads per instance for scanning the index</td>
</tr>
<tr>
<td>INSTANCES</td>
<td>Number of instances across which the index is to be scanned</td>
</tr>
<tr>
<td>PARTITIONED</td>
<td>Whether the index is partitioned</td>
</tr>
<tr>
<td>TEMPORARY</td>
<td>Whether the index is on a temporary table</td>
</tr>
<tr>
<td>GENERATED</td>
<td>Whether the name of the index is system generated</td>
</tr>
<tr>
<td>SECONDARY</td>
<td>N/A</td>
</tr>
<tr>
<td>BUFFER_POOL</td>
<td>Whether the index is a secondary object</td>
</tr>
<tr>
<td>USER_STATS</td>
<td>N/A</td>
</tr>
<tr>
<td>DURATION</td>
<td>N/A</td>
</tr>
<tr>
<td>PCT_DIRECT_ACCESS</td>
<td>N/A</td>
</tr>
<tr>
<td>ITYP_OWNER</td>
<td>N/A</td>
</tr>
<tr>
<td>ITYP_NAME</td>
<td>N/A</td>
</tr>
<tr>
<td>PARAMETERS</td>
<td>N/A</td>
</tr>
<tr>
<td>GLOBAL_STATS</td>
<td>N/A</td>
</tr>
<tr>
<td>DOMIDX_STATUS</td>
<td>N/A</td>
</tr>
<tr>
<td>DOMIDX_OPSTATUS</td>
<td>N/A</td>
</tr>
<tr>
<td>FUNCIDX_STATUS</td>
<td>N/A</td>
</tr>
<tr>
<td>JOIN_INDEX</td>
<td>N/A</td>
</tr>
<tr>
<td>IOT_REDUNDANT_PKEY_ELIM</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**A.24 USER_OBJECTS**

The **USER_OBJECTS** view contains objects owned by the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJECT_NAME</td>
<td>Name of the object</td>
</tr>
</tbody>
</table>
### A.25 USER_SYNONYMS

The **USER_SYNONYMS** view contains the private synonyms of the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNONYM_NAME</td>
<td>Name of the synonym</td>
</tr>
<tr>
<td>TABLE_OWNER</td>
<td>Owner of the object referenced by the synonym</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Name of the object referenced by the synonym</td>
</tr>
<tr>
<td>DB_LINK</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### A.26 USER_TABLES

The **USER_TABLES** view contains description of the tables owned by the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_NAME</td>
<td>Name of the table</td>
</tr>
<tr>
<td>TABLESPACE_NAME</td>
<td>Name of the tablespace containing the table</td>
</tr>
<tr>
<td>CLUSTER_NAME</td>
<td>N/A</td>
</tr>
<tr>
<td>IOT_NAME</td>
<td>Name of the index organized table</td>
</tr>
<tr>
<td>PCT_FREE</td>
<td>N/A</td>
</tr>
<tr>
<td>PCT_USED</td>
<td>N/A</td>
</tr>
<tr>
<td>INIT_TRANS</td>
<td>N/A</td>
</tr>
<tr>
<td>MAX_TRANS</td>
<td>N/A</td>
</tr>
<tr>
<td>INITIAL_EXTENT</td>
<td>N/A</td>
</tr>
<tr>
<td>NEXT_EXTENT</td>
<td>N/A</td>
</tr>
<tr>
<td>MIN_EXTENTS</td>
<td>N/A</td>
</tr>
<tr>
<td>Column name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MAX_EXTENTS</td>
<td>N/A</td>
</tr>
<tr>
<td>PCT_INCREASE</td>
<td>N/A</td>
</tr>
<tr>
<td>FREELISTS</td>
<td>Number of process freelists allocated to a segment</td>
</tr>
<tr>
<td>FREELIST_GROUPS</td>
<td>Number of freelist groups allocated to a segment</td>
</tr>
<tr>
<td>LOGGING</td>
<td>Logging information</td>
</tr>
<tr>
<td>BACKED_UP</td>
<td>N/A</td>
</tr>
<tr>
<td>NUM_ROWS</td>
<td>Number of rows in the table</td>
</tr>
<tr>
<td>BLOCKS</td>
<td>N/A</td>
</tr>
<tr>
<td>EMPTY_BLOCKS</td>
<td>N/A</td>
</tr>
<tr>
<td>AVG_SPACE</td>
<td>N/A</td>
</tr>
<tr>
<td>CHAIN_CNT</td>
<td>N/A</td>
</tr>
<tr>
<td>AVG_ROW_LEN</td>
<td>Average length of a row in the table in bytes</td>
</tr>
<tr>
<td>AVG_SPACE_FREELIST_BLOCKS</td>
<td>Average freespace of all blocks on a freelist</td>
</tr>
<tr>
<td>NUM_FREELIST_BLOCKS</td>
<td>Number of blocks on the freelist</td>
</tr>
<tr>
<td>DEGREE</td>
<td>Number of threads per instance for scanning the table</td>
</tr>
<tr>
<td>Instances</td>
<td>Number of instances across which the table is to be scanned</td>
</tr>
<tr>
<td>CACHE</td>
<td>Whether the cluster is to be cached in the buffer cache</td>
</tr>
<tr>
<td>TABLE_LOCK</td>
<td>Whether table locking is enabled or disabled</td>
</tr>
<tr>
<td>SAMPLE_SIZE</td>
<td>Sample size used in analyzing this table</td>
</tr>
<tr>
<td>LAST_ANALYZED</td>
<td>Date on which this table was most recently analyzed</td>
</tr>
<tr>
<td>PARTITIONED</td>
<td>Indicates whether this table is partitioned</td>
</tr>
<tr>
<td>IOT_TYPE</td>
<td>If this is an index organized table</td>
</tr>
<tr>
<td>TEMPORARY</td>
<td>Can the current session only see data that it placed in this object itself?</td>
</tr>
<tr>
<td>SECONDARY</td>
<td>N/A</td>
</tr>
<tr>
<td>NESTED</td>
<td>If the table is a nested table</td>
</tr>
<tr>
<td>BUFFER_POOL</td>
<td>The default buffer pool for the object</td>
</tr>
<tr>
<td>ROW_MOVEMENT</td>
<td>N/A</td>
</tr>
<tr>
<td>GLOBAL_STATS</td>
<td>N/A</td>
</tr>
<tr>
<td>USER_STATS</td>
<td>N/A</td>
</tr>
<tr>
<td>DURATION</td>
<td>N/A</td>
</tr>
<tr>
<td>SKIP_CORRUPT</td>
<td>N/A</td>
</tr>
<tr>
<td>MONITORING</td>
<td>N/A</td>
</tr>
<tr>
<td>CLUSTER_OWNER</td>
<td>N/A</td>
</tr>
<tr>
<td>DEPENDENCIES</td>
<td>N/A</td>
</tr>
<tr>
<td>COMPRESSION</td>
<td>N/A</td>
</tr>
</tbody>
</table>
**A.27 USER_TAB_COLUMNS**

The `USER_TAB_COLUMNS` view contains columns of the tables, views, and clusters owned by the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_NAME</td>
<td>Name of the table, view, or cluster</td>
</tr>
<tr>
<td>COLUMN_NAME</td>
<td>Name of the column</td>
</tr>
<tr>
<td>DATA_TYPE</td>
<td>Data type of column</td>
</tr>
<tr>
<td>DATA_TYPE_MOD</td>
<td>Data type modifier of the column</td>
</tr>
<tr>
<td>DATA_TYPE_OWNER</td>
<td>Owner of the data type of the column</td>
</tr>
<tr>
<td>DATA_LENGTH</td>
<td>Maximum length of the column in bytes</td>
</tr>
<tr>
<td>DATA_PRECISION</td>
<td>N/A</td>
</tr>
<tr>
<td>DATA_SCALE</td>
<td>Digits to the right of a decimal point in a number</td>
</tr>
<tr>
<td>NULLABLE</td>
<td>Whether the column permits nulls. Value is <code>n</code> if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY key.</td>
</tr>
<tr>
<td>COLUMN_ID</td>
<td>Sequence number of the column as created</td>
</tr>
<tr>
<td>DEFAULT_LENGTH</td>
<td>N/A</td>
</tr>
<tr>
<td>DATA_DEFAULT</td>
<td>N/A</td>
</tr>
<tr>
<td>NUM_DISTINCT</td>
<td>Number of distinct values in each column of the table</td>
</tr>
<tr>
<td>LOW_VALUE</td>
<td>For tables with more than three rows, the second lowest and second highest values. These statistics are expressed in hexadecimal notation for the internal representation of the first 32 bytes of the values.</td>
</tr>
<tr>
<td>HIGH_VALUE</td>
<td>N/A</td>
</tr>
<tr>
<td>DENSITY</td>
<td>N/A</td>
</tr>
<tr>
<td>NUM_NULLS</td>
<td>Number of nulls in the column</td>
</tr>
<tr>
<td>NUM_BUCKETS</td>
<td>Number of buckets in histogram for the column</td>
</tr>
<tr>
<td>LAST_ANALYZED</td>
<td>Date on which this column was most recently analyzed</td>
</tr>
<tr>
<td>SAMPLE_SIZE</td>
<td>Sample size used in analyzing this column</td>
</tr>
<tr>
<td>CHARACTER_SET_NAME</td>
<td>Name of the character set</td>
</tr>
<tr>
<td>CHAR_COL_DECL_LENGTH</td>
<td>Length of the character set</td>
</tr>
<tr>
<td>GLOBAL_STATS</td>
<td>N/A</td>
</tr>
<tr>
<td>USER_STATS</td>
<td>N/A</td>
</tr>
<tr>
<td>AVG_COL_LEN</td>
<td>Average length of the column (in bytes)</td>
</tr>
<tr>
<td>CHAR_LENGTH</td>
<td>Length of the column in characters</td>
</tr>
<tr>
<td>CHAR_USED</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**A.28 USER_TAB_COMMENTS**

The `USER_TAB_COMMENTS` view contains comments on the tables and views owned by the user.
### A.29 USER_VIEWS

The `USER_VIEWS` view contains text of views owned by the user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_NAME</td>
<td>Name of the object</td>
</tr>
<tr>
<td>TABLE_TYPE</td>
<td>Type of the object</td>
</tr>
<tr>
<td>COMMENTS</td>
<td>Comments on the object</td>
</tr>
</tbody>
</table>

### A.30 USER_USERS

The `USER_USERS` view contains information about the current user.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USERNAME</td>
<td>Name of the user</td>
</tr>
<tr>
<td>USER_ID</td>
<td>N/A</td>
</tr>
<tr>
<td>ACCOUNT_STATUS</td>
<td>Indicates if the account is locked, expired or unlocked</td>
</tr>
<tr>
<td>LOCK_DATE</td>
<td>Date on which the account was locked</td>
</tr>
<tr>
<td>EXPIRE_DATE</td>
<td>Date of expiration of the account</td>
</tr>
<tr>
<td>DEFAULT_TABLESPACE</td>
<td>N/A</td>
</tr>
<tr>
<td>TEMPORARY_TABLESPACE</td>
<td>N/A</td>
</tr>
<tr>
<td>CREATED</td>
<td>N/A</td>
</tr>
<tr>
<td>EXTERNAL_NAME</td>
<td>Name of the external user</td>
</tr>
</tbody>
</table>
B

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. Changes made to the initialization parameters only take effect in the next gateway session.

The following sections contain 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.

• Initialization Parameter File Syntax
• Oracle Database Gateway for DRDA Initialization Parameters

B.1 Initialization Parameter File Syntax

The syntax for the initialization parameter file is as follows:

• The file is a sequence of commands.
• Each command should start on a separate line.
• End of line is considered a command terminator (unless escaped with a backslash).
• If there is a syntax error in an initialization parameter file, none of the settings take effect.
• 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:

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

  2. 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
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 username 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).

B.2 Oracle Database Gateway for DRDA Initialization Parameters

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

- HS_CALL_NAME
- 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
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>None</td>
</tr>
<tr>
<td>Range of values</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
Specifies the remote functions that can be referenced in SQL statements. The value is a list of remote functions and their owners, separated by semicolons, in the following format:

\[owner\_name.\]function\_name

For example:

owner1.A1;owner2.A2;A3

If an owner name is not specified for a remote function, the default owner name becomes the user name used to connect to the remote database (specified when the Heterogeneous Services database link is created or taken from user session if not specified in the DB link).

The entries for the owner names and the function names are case-sensitive.

### B.4 HS_DB_DOMAIN

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>WORLD</td>
</tr>
<tr>
<td>Range of values</td>
<td>1 to 199 characters</td>
</tr>
</tbody>
</table>

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.

### B.5 HS_DB_INTERNAL_NAME

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>01010101</td>
</tr>
<tr>
<td>Range of values</td>
<td>1 to 16 hexadecimal characters</td>
</tr>
</tbody>
</table>

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.
B.6 HS_DB_NAME

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>HO</td>
</tr>
<tr>
<td>Range of values</td>
<td>1 to 8 characters</td>
</tr>
</tbody>
</table>

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.

B.7 HS_DESCRIBE_CACHE_HWM

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>100</td>
</tr>
<tr>
<td>Range of values</td>
<td>1 to 4000</td>
</tr>
</tbody>
</table>

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.

B.8 HS_LANGUAGE

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>System-specific</td>
</tr>
<tr>
<td>Range of values</td>
<td>Any valid language name (up to 255 characters)</td>
</tr>
</tbody>
</table>

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.
B.8.1 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.

B.8.2 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.

B.8.3 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.

B.9 HS_LONG_PIECE_TRANSFER_SIZE

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>64 KB</td>
</tr>
<tr>
<td>Range of values</td>
<td>Any value up to 2 GB</td>
</tr>
</tbody>
</table>
Sets the size of the piece of \texttt{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.

### B.10 HS\_OPEN\_CURSORS

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>50</td>
</tr>
<tr>
<td>Range of values</td>
<td>1 to the value of \texttt{OPEN_CURSORS} initialization parameter of Oracle database</td>
</tr>
</tbody>
</table>

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 \texttt{OPEN\_CURSORS} initialization parameter in the Oracle database is recommended.

### B.11 HS\_RPC\_FETCH\_REBLOCKING

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>ON</td>
</tr>
<tr>
<td>Range of values</td>
<td>OFF or ON</td>
</tr>
</tbody>
</table>

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:

- \texttt{OFF} disables reblocking of fetched data so that data is immediately sent from agent to server.
- \texttt{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 \texttt{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.

### B.12 HS\_RPC\_FETCH\_SIZE

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>50000</td>
</tr>
<tr>
<td>Range of values</td>
<td>1 to 10000000</td>
</tr>
</tbody>
</table>
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.

**B.13 HS_TRANSACTION_MODEL**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>COMMIT_CONFIRM</td>
</tr>
<tr>
<td>Range of Values</td>
<td>COMMIT_CONFIRM, READ_ONLY, READ_ONLY_AUTOCOMMIT, SINGLE_SITE, SINGLE_SITE_AUTOCOMMIT</td>
</tr>
</tbody>
</table>

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.

  COMMIT_CONFIRM does not apply to Oracle Database Gateway for ODBC. The default value for Oracle Database Gateway for ODBC is SINGLE_SITE.

- **READ_ONLY** provides read access to the non-Oracle database.

- **READ_ONLY_AUTOCOMMIT** provides read access to the non-Oracle database that do not have logging.

- **SINGLE_SITE** provides read and write access to the non-Oracle database. However, the gateway cannot participate in distributed updates.

- **SINGLE_SITE_AUTOCOMMIT** provides read and write access to the non-Oracle database that do not have logging. Any update is committed immediately, and the gateway cannot participate in distributed updates.
**B.14 IFILE**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>None</td>
</tr>
<tr>
<td>Range of values</td>
<td>Valid parameter file names</td>
</tr>
</tbody>
</table>

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*

**B.15 HS_FDS_CONNECT_INFO**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>None</td>
</tr>
<tr>
<td>Range of Values</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

`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 gateways are as follows:

`HS_FDS_CONNECT_INFO=IP_address:Port_number/Database_name,Type`

*Where IP_address is the hostname or IP address of the DB2 DRDA server*

*Port_number is the port number of the DB2 DRDA server*

*Database_name is the database name of the DB2 server*

*Type (case insensitive) is one of the following:*

- ZOS (DB2 UDB for z/OS),
- IOS (DB2 UDB for iSeries), or
- LUW (DB2 UDB for Linux, Unix, or Windows)

This release of gateway can support IPv6. If IPv6 address format is to be specified, you would need to wrap square brackets around the IPv6 specification to indicate the separation from the port number.

For example,

`HS_FDS_CONNECT_INFO=[2001:0db8:20C:F1FF:FEC6:38AF]:1300/DB2M,ZOS`
### B.16 HS_FDS_RECOVERY_ACCOUNT

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>RECOVER</td>
</tr>
<tr>
<td>Range of values</td>
<td>Any valid user ID</td>
</tr>
</tbody>
</table>

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.

For DRDA, `HS_FDS_RECOVERY_ACCOUNT` specifies the user ID that is used by the gateway if a distributed transaction becomes in doubt. This user ID must have execute privileges on the package and must be defined to the IBM database.

If a distributed transaction becomes in doubt, then the Oracle database determines the status of the transaction by connecting to the IBM database, using the `HS_FDS_RECOVERY_ACCOUNT`. If this parameter is missing, then the gateway attempts to connect to a user ID of `ORARECOV`.

The name of the recovery account is case-sensitive.

### B.17 HS_FDS_RECOVERY_PWD

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>RECOVER</td>
</tr>
<tr>
<td>Range of values</td>
<td>Any valid password</td>
</tr>
</tbody>
</table>

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.

`HS_FDS_RECOVERY_PWD` is used with the `HS_FDS_RECOVERY_ACCOUNT`. The recovery user connects to the IBM database if a distributed transaction is in doubt.

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

### B.18 HS_FDS_FETCH_ROWS

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>100</td>
</tr>
<tr>
<td>Range of Values</td>
<td>Any integer between 1 and 1000</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>HS_FDS_FETCH_ROWS=num</code></td>
</tr>
</tbody>
</table>

`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.
**B.19 HS_FDS_TRACE_LEVEL**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>OFF</td>
</tr>
<tr>
<td>Range of values</td>
<td>OFF, ON, DEBUG</td>
</tr>
</tbody>
</table>

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.

**B.20 HS_FDS_TRANSACTION_LOG**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>HS_TRANSACTION_LOG</td>
</tr>
<tr>
<td>Range of Values</td>
<td>Any valid table name</td>
</tr>
</tbody>
</table>

规定的名称的表创建在非Oracle系统中用于记录交易。有关交易模型的更多信息，请参见 **HS_TRANSACTION_MODEL** 参数。

**B.21 HS_IDLE_TIMEOUT**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>0 (no timeout)</td>
</tr>
<tr>
<td>Range of Values</td>
<td>0-9999 (minutes)</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_IDLE_TIMEOUT=num</td>
</tr>
</tbody>
</table>

此功能仅适用于Oracle Net TCP协议。

当没有活动的连接网关会话的指定时间期，网关会话将被自动终止。如果更新（如有）仍然存在则回滚。

**B.22 HS_FDS_MBCS_TO_GRAPHIC**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>FALSE</td>
</tr>
</tbody>
</table>
### B.23 HS_FDS_GRAPHIC_TO_MBCS

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>FALSE</td>
</tr>
<tr>
<td>Range of Values</td>
<td>FALSE</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_MBCS_TO_GRAPHIC= {FALSE</td>
</tr>
</tbody>
</table>

If set to **TRUE**, any single-byte character meant to insert to DB2 (var)graphic column would be converted to equivalent double-byte value before the insert operation.

### B.24 HS_FDS_TIMESTAMP_MAPPING

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>CHAR</td>
</tr>
<tr>
<td>Range of Values</td>
<td>CHAR</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_TIMESTAMP_MAPPING= {CHAR</td>
</tr>
</tbody>
</table>

If set to **CHAR**, then non-Oracle target timestamp would be mapped to **CHAR(26)**. If set to **DATE** (default), then non-Oracle target timestamp would be mapped to Oracle **DATE**. If set to **TIMESTAMP**, then non-Oracle target timestamp would be mapped to Oracle **TIMESTAMP**.

### B.25 HS_FDS_DATE_MAPPING

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>DATE</td>
</tr>
<tr>
<td>Range of Values</td>
<td>DATE</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_DATE_MAPPING= {DATE</td>
</tr>
</tbody>
</table>

If set to **CHAR**, then non-oracle target date would be mapped to **CHAR(10)**. If set to **DATE**, then non-Oracle target date would be mapped to Oracle date.
### B.26 HS_FDS_QUOTE_IDENTIFIER

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>TRUE</td>
</tr>
<tr>
<td>Range of Values</td>
<td>TRUE</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_QUOTE_IDENTIFIER={FALSE</td>
</tr>
</tbody>
</table>

By default, the gateway will quote identifiers if the FDS supports it. However, we give the user the ability to overwrite the behavior.

### B.27 HS_FDS_CAPABILITY

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>None</td>
</tr>
<tr>
<td>Range of Values</td>
<td>Refer to <a href="#">SQL Functions That Can Be Enabled</a></td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_CAPABILITY= {FUNCTION/{ON</td>
</tr>
</tbody>
</table>

If the `HS_FDS_CAPABILITY` is set to `ON` then the specified function will be sent to DB2 for processing. In other words, post processing will be not needed for that function.

If the `HS_FDS_CAPABILITY` is set to `OFF` then the specified function will be not be sent to DB2 for processing. In other words, it will be post processed.

If the `HS_FDS_CAPABILITY` is set to `SKIP` then the specified function will be stripped from the SQL statement sent to DB2. In other words the function will be ignored.

### B.28 HS_FDS_TRANSACTION_ISOLATION

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>READ_COMMITTED</td>
</tr>
<tr>
<td>Range of Values</td>
<td>{READ_UNCOMMITTED</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_ISOLATION_LEVEL={(READ_UNCOMMITTED</td>
</tr>
</tbody>
</table>

`HS_FDS_TRANSACTION_ISOLATION` specifies the isolation level that is used for the transaction that the gateway opens on the non-Oracle database.

The isolation level `NONE` is only valid against DB2 iSeries. It specifies level `NO COMMIT`. This isolation level will force an `HS_TRANSACTION_MODEL` of `SINGLE_SITE_AUTOCOMMIT`. Please refer to DB2's documentation regarding this level.

The isolation levels of `READ_UNCOMMITTED`, `READ_COMMITTED`, `REPEATABLE_READ`, and `SERIALIZABLE` are the four isolation levels defined in the SQL standard and adopted by both both ANSI and ISO/IEC. For additional information regarding them, see *Oracle Database Concepts*.
Use caution when specifying an isolation level lower than the Oracle transaction isolation level being used, as the gateway transaction will have different Preventable Read Phenomena from what will occur in the Oracle database transaction.

### B.29 HS_FDS_PACKAGE_COLLID

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>ORACLEGTW</td>
</tr>
<tr>
<td>Range of Values</td>
<td>An alphanumeric string 1 to 18 characters in length</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_PACKAGE_COLLID=collection_id</td>
</tr>
</tbody>
</table>

**Note:**

Any change to this parameter will cause a new package to be implicitly bound by the gateway. For DB2 for UDB iSeries, prior to attempting a connection, one should use the iSeries SQL command `CREATE SCHEMA` or `CREATE COLLECTION` to create an iSeries library with the name as specified for `HS_FDS_PACKAGE_COLLID`. This `COLLECTION` or `SCHEMA` should be created under the id specified in the `CONNECT TO` phrase of the Oracle SQL command `CREATE DATABASE LINK`.

### B.30 HS_NLS_LENGTH_SEMANTICS

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>BYTE</td>
</tr>
<tr>
<td>Range of Values</td>
<td>BYTE</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_NLS_LENGTH_SEMANTICS = { BYTE</td>
</tr>
</tbody>
</table>

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 UDB database 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.

### B.31 HS_KEEP_REMOTE_COLUMN_SIZE

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>OFF</td>
</tr>
<tr>
<td>Range of Values</td>
<td>OFF</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_KEEP_REMOTE_COLUMN_SIZE = OFF</td>
</tr>
</tbody>
</table>
**Property** | **Description**
--- | ---
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 UDB 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 to the gateway. When it is set to LOCAL, the expansion is suppressed between the gateway and 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 the non-Oracle database to 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.

---

**B.32 HS_FDS_RESULTSET_SUPPORT**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>FALSE</td>
</tr>
<tr>
<td>Range of values</td>
<td>TRUE</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_RESULTSET_SUPPORT = { TRUE</td>
</tr>
</tbody>
</table>

Enables result sets to be returned from stored procedures. By default, all stored procedures do not return a result set to the user.

**Note:**

If you set this initialization parameter, you must do the following:

- Change the syntax of the procedure execute statement for all existing stored procedures, to handle result sets.
- Work in the sequential mode of Heterogeneous Services.

---

**B.33 HS_FDS_REMOTE_DB_CHARSET**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>None</td>
</tr>
<tr>
<td>Range of values</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDSREMOTE_DB_CHARSET</td>
</tr>
</tbody>
</table>
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`.

### B.34 HS_FDS_SUPPORT_STATISTICS

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>TRUE</td>
</tr>
<tr>
<td>Range of values</td>
<td>`{TRUE</td>
</tr>
<tr>
<td>Syntax</td>
<td>`HS_FDS_SUPPORT_STATISTICS= {TRUE</td>
</tr>
</tbody>
</table>

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`.

### B.35 HS_FDS_RSET_RETURN_ROWCOUNT

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>FALSE</td>
</tr>
<tr>
<td>Range of values</td>
<td>`{TRUE</td>
</tr>
<tr>
<td>Syntax</td>
<td>`HS_FDS_RSET_RETURN_ROWCOUNT= {TRUE</td>
</tr>
</tbody>
</table>

When set to `TRUE`, the gateway returns the row counts of DML statements that are executed inside a stored procedure. The row count is returned as a single row, single column result set of type signed integer.

When set to `FALSE`, the gateway skips the row counts of DML statements that are executed inside a stored procedure. This is the default behavior, and it is the behavior of 11.1 and older gateways.

### B.36 HS_FDS_AUTHENTICATE_USER

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>CLEARTEXT</td>
</tr>
<tr>
<td>Range of values</td>
<td>`{CLEARTEXT</td>
</tr>
<tr>
<td>Syntax</td>
<td>`HS_FDS_AUTHENTICATE_USER= {CLEARTEXT</td>
</tr>
</tbody>
</table>

Specifies the way in which user ID and password are sent to the remote DB2 server and authenticated. Valid values are:
• **CLEARTEXT**: user ID and password are sent in clear text to server (default).
• **ENCRYPT**: password is sent encrypted to server.
• **ENCRYPT_BOTH**: user ID and password are sent encrypted to server.
• **CLIENT**: user ID is validated on the client side instead of by the server.
• **KERBEROS**: uses Kerberos to authenticate user ID.

### B.37 HS_FDS_ENCRYPT_SESSION

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>NONE</td>
</tr>
<tr>
<td>Range of values</td>
<td>(NONE</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_ENCRYPT_SESSION = {NONE</td>
</tr>
</tbody>
</table>

Specifies the way the session to DB2 is encrypted. Valid values are:

- **NONE**: data session is not encrypted (default).
- **SSL**: Use SSL to encrypt data session.
- **DB2**: Use DB2 encryption protocol for data session (supported only on DB2 for Linux, UNIX, Windows, and DB2 for z/OS).
- **NOTRUST_SSL**: This option is equivalent to the SSL setting, with initialization parameter `HS_FDS_VALIDATE_SERVER_CERT = DISABLED`.

### B.38 HS_FDS_VALIDATE_SERVER_CERT

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>ENABLED</td>
</tr>
<tr>
<td>Range of values</td>
<td>(ENABLED</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_VALIDATE_SERVER_CERT = (ENABLED</td>
</tr>
</tbody>
</table>

Specifies whether the driver validates the certificate that is sent by the database server when SSL encryption is enabled through `HS_FDS_ENCRYPT_SESSION`. When using SSL server authentication, any certificate sent by the server must be issued by a trusted Certificate Authority. Valid values are:

- **ENABLED**: the gateway validates the certificate that is sent by the database server. Any certificate from the server must be issued by a trusted Certificate Authority in the truststore file. The truststore information is specified using the `HS_FDS_TRUSTSTORE_FILE` and `HS_FDS_TRUSTSTORE_PASSWORD` initialization parameters.
- **DISABLED**: the gateway does not validate the certificate that is sent by the database server.
B.39 HS_FDS_TRUSTSTORE_FILE

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>none</td>
</tr>
<tr>
<td>Range of values</td>
<td>path to truststore file</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_TRUSTSTORE_FILE = path to truststore file</td>
</tr>
</tbody>
</table>

Specifies the path that specifies the location of the truststore file. The truststore file contains a list of the valid Certificate Authorities (CAs) that are trusted by the client machine for SSL server authentication.

B.40 HS_FDS_TRUSTSTORE_PASSWORD

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>none</td>
</tr>
<tr>
<td>Range of values</td>
<td>password</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_TRUSTSTORE_PASSWORD = password</td>
</tr>
</tbody>
</table>

Specifies the password required to access the truststore.

B.41 HS_FDS_SQLLEN_INTERPRETATION

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>64</td>
</tr>
<tr>
<td>Range of values</td>
<td>{64, 32}</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_SQLLEN_INTERPRETATION = {64, 32}</td>
</tr>
</tbody>
</table>

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.

B.42 HS_FDS_ARRAY_EXEC

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Value</td>
<td>FALSE</td>
</tr>
<tr>
<td>Range of values</td>
<td>{TRUE, FALSE}</td>
</tr>
<tr>
<td>Syntax</td>
<td>HS_FDS_ARRAY_EXEC = {TRUE, FALSE}</td>
</tr>
</tbody>
</table>
If set to **TRUE**, the gateway will use array operations for insert, update, delete statements containing binds against the remote data source. The array size is determined by the value of the `HS_FDS_FETCH_ROWS` init parameter.

If set to **FALSE**, the gateway will not use array operations for insert, update, and delete statements. Instead, a single statement will be issued for every value.
Index

A

accessing gateway
    main topic, 3-4
Advanced Security
    function of the gateway, 1-4
    purpose, 1-4
ALL_CATALOG view, A-2
ALL_COL_COMMENTS view, A-2
ALL_CON_COLUMNS view, A-2
ALL_CONSTRAINTS view, A-2
ALL_DB_LINKS data dictionary view, 3-4
ALL_INDEXES view, A-3
ALL_OBJECTS view, A-5
ALL_SYNONYMS view, A-6
ALL_TAB_COMMENTS view, A-8
ALL_TABLES view, A-6
ALL_USERS view, A-9
ALL_VIEWS view, A-9
ALTER SESSION statement, 3-3
ANSI-standard SQL, 1-4, 1-12
APPEND command
    supported by COPY, 3-8
application
    portability, 1-10
    server support, 1-3
application development on the gateway, 1-13
architecture of the gateway, 1-7
array size
    fetch reblocking, 1-11
    how determined, 4-2
AS/400
    files and file members, accessing, 3-5
library name, HS_FDS_PACKAGE_COLLID, B-14
ASCII
    sort order, 4-21
    translated from EBCDIC, 4-23
autonomy, site, 1-6

B

binary data, non-character, 4-23
bind variables
    SQL passthrough, 4-28
bug
    debugging
        drc values in the DRDA software, 5-2
        setting trace parameters, 5-3
        SQL tracing, 3-8
        number 205538, known restrictions, SQL limitations, 2-6

C

call
    DB2 stored procedure, 4-3, 4-5
    empproc stored procedure, 4-4
    Oracle Call Interfaces, 5-1
    PL/SQL, 4-4
stored procedure
    creating a synonym to maintain location transparency, 4-3
    using standard Oracle PL/SQL, 4-2
to stored procedure
    known restrictions, 2-4
capabilities of DRDA server, native semantics, 4-19
changes in this release
    IBM DB2/UDB supported, 1-13
    read-only support, 1-10
character sets
    Heterogeneous Services, B-6
character string
    converting datatypes, 4-23
    performing operations, 4-23
CHECKSUM command
    extended advanced networking, 1-4
clauses
    CONNECT TO, 3-2
    GROUP BY, SQL Set Clauses, 4-21
    HAVING, SQL Set Clauses, 4-21
    ORDER BY, SQL Set Clauses, 4-21
    SQL
        DELETE, 4-26
        INSERT, 4-26
        SELECT WHERE, 4-26
        UPDATE, 4-26
clauses (continued)
USING, 3-2
VALUES
functions not allowed by DB2, 4-26
WHERE
known restrictions, SQL limitations, 2-6
SQL Set Clauses, 4-21
WHERE CURRENT OF CURSOR,
known restrictions, SQL
limitations, 2-6
closing and opening again any session against
db2 required with any change to
HS_FDS_PACKAGE_COLLID, B-14
corercion
of data, 4-19
column
date columns, TO_DATE function, 4-26
supported in a result set, 1-10
commands
CHECKSUM, 1-4
COPY
Oracle database to DRDA server, 3-7
SQL*Plus command, 3-8
EXECUTE, 1-5
EXPLAIN PLAN, 5-3
commit confirm protocol, 1-6
compatible SQL set operators and clauses, 4-21
CONNECT TO clause, 3-2
convert
character string, 4-23
datatypes
DRDA to Oracle datatypes, 4-21
DATE, 4-24
floating point to integer, 4-27
into most suitable datatype, 4-27
SQL, 1-9
to the numeric datatype, 4-27
converter, protocol, 1-4
COPY command
Oracle database to DRDA server, 3-7
COPY SQL*Plus command, 3-8
copying data
from the DRDA server, 3-8
from the Oracle database to DRDA server, 3-7
COS SQL function, 4-10
COUNT function, 4-27
CREATE command
supported by COPY, 3-8
CREATE DATABASE LINK command, 3-2
CREATE TABLE statement, 1-5
creating
database link, 3-2

D

data coercion, 4-19
data control language (DCL), 1-5
DATA datatype, 4-24
data definition language (DDL), 1-4
data dictionary
using, 4-30
views
ALL_DB_LINKS, 3-4
eulation on DRDA server, 4-30
for DB2/UDB not supported, A-1
supported for DB2/OS390 and DB2/400
servers, A-1
USER_DB_LINKS, 3-4
database
catalogs, 4-30
link
behavior, 4-9
creating, 3-2
dropping links, 3-3
examining, 3-4
guidelines, 3-2
limits, 3-4
processing, 3-2
suffix, 4-1
to identify the gateway, 1-9
triggers, 1-3
datatype
character string, 4-22
column (ALL_TAB_COLUMNS), A-8
column (USER_TAB_COLUMNS), A-16
conversion
DRDA to Oracle datatypes, 4-21
no control over, 4-27
converting character string, 4-23
data and time, 4-24
differences between Oracle database and
DRDA databases, 4-21
DRDA server datatypes list, 4-22
mapping, 4-22
numeric, 4-27
operations, numeric, 4-27
Oracle datatypes RAW and LONG RAW,
4-23
restrictions, 4-22
size and value limitations, 4-22
datatypes
DATE, 4-24
GRAPHIC, 4-23
LONG RAW, 4-23
Oracle and IBM DATE, 4-24
Oracle DATE, 4-24
RAW
character string operations, 4-23
datatypes (continued)
  TIME, 4-24
  TIMESTAMP, 4-24
date
  date columns, TO_DATE function, 4-26
  HS_NLS_DATE_FORMAT parameter, 4-26
INSERT
  statement, 4-26
  operations, 4-24
SELECT statement, 4-26
TO_DATE function, 4-26
UPDATE
  statement, 4-26
date arithmetic
  known restrictions, 2-5
DATE datatype, 4-24
DB_DOMAIN parameter
  known restrictions, 2-4
DB2
  data access, 1-5
  native SQL, 1-5
  native stored procedures, 1-5
  procedural feature considerations, 4-5
  SQL statements, 4-29
  statements
    CREATE TABLE, 1-5
  stored procedures, 4-5
DB2 UDB for iSeries
  HS_FDS_PACKAGE_COLLID, B-14
DB2/400
  catalog view, 4-30
  data dictionary views supported by gateway, A-1
DB2/OS390
  catalog view, 4-30
  data dictionary views supported by gateway, A-1
  V6, V7 and V8 stored procedures supported, 1-10
DB2/UDB
  catalog view, 4-30
  data dictionary views not supported, A-1
  supported, 1-13
DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE
  DATE function, 4-28
DD basic tables, known restrictions, 2-4
DDL
  statement, 4-28
debug
  gateway, 5-3
debugging
  error codes, 5-2
  SQL tracing, 5-3
  your application, 3-8
DELETE (continued)
  known restrictions, SQL limitations, 2-6
  operation, 4-1
  SQL clause, 4-26
  statement, 4-28
DESCRIBE
  character string operations, 4-23
  describe cache high water mark
    definition, B-5
dictionary
  mapping, 1-4
  tables, 4-30
DICTIONARY view, A-10
distributed
  applications, support for, 1-10
  DRDA transactions, 3-7
  queries
    two-phase commit, 3-6
  transaction, HS_FDS_RECOVERY_ACCOUNT, B-10
DRDA
  catalog, 4-30
  DRDA server
    architecture, 1-8
    capabilities, native semantics, 4-19
    functions, 4-19
    stored procedures, 4-3
  DRDA server error, 5-2
  DROP DATABASE LINK statement, 3-3
  dynamic dictionary mapping, 1-4
E
  EBCDIC
    sort order, 4-21
    translated to ASCII, 4-23
EMP
  system-wide synonym, 3-5
  environment
    heterogeneous, 3-7
errmc
  errmc field lists any error tokens, 5-2
error
  basic description, 5-1
  detected
    by Oracle database, 5-1
    by server database, 5-2
    in DRDA software, 5-2
  interpreting error messages, 5-1
  messages
    Oracle LONG datatype is too long, 4-23
  messages ((amp)) codes, 5-1
  ORA-02019, 5-1
  ORA-28500 (was ORA-09100), 5-2
error (continued)
  Oracle mapped error codes, 5-3
tokens, 5-2
translation, 4-23
  with Native Semantics, 4-19
Error messages
  error tracing, B-11
EXCEPT set operator, SQL Set Clauses, 4-21
EXECUTE command, 1-5
exits
  gateway local date, 4-26
EXPLAIN PLAN command, 5-3

F

features of the gateway
  application development and end-user tools, 1-13
  application portability, 1-10
columns supported in a result set, 1-10
distributed applications supported, 1-10
EXPLAIN_PLAN improvement, 1-10
fetch reblocking, 1-10
heterogeneous database integration, 1-10
heterogeneous services architecture, 1-10
main topic, 1-10
minimum impact on existing systems, 1-10
Native Semantics, 1-10
Oracle database passthrough supported, 1-10
Oracle snapshots, 3-7
performance enhancements, 1-10
remote data access, 1-10
retrieving result sets through passthrough, 1-10
  support for TCP/IP, 1-10
fetch array size, with HS_FDS_Fetch_ROWS, B-10
fetch reblocking
  controlled by two Heterogeneous Services
  initialization parameters, 1-10
  supported by Oracle database, 4-2
fields
  errmc, lists any error tokens, 5-2
file member
  accessing AS/400 files, 3-5
  name, 3-5
FOR BIT DATA
  option, 4-23
functions
  COS, 4-10
  COUNT, 4-27
  DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE, 4-28
  DRDA server, 4-19

G

gateway
  accessing
    main topic, 3-4
  advantages
    migration and coexistence, 1-6
    multi-site transactions, 1-6
    security, 1-6
    server technology and tools, 1-5
    site autonomy, 1-6
    two-phase commit, 1-6
  and Oracle tools, 1-9
  and stored procedures (Oracle and non-Oracle), 1-5
  application tools, 1-13
  architecture, 1-7
  benefits of integration with Oracle database, 1-3
  components, 1-8
  definition of terms, 1-7
  features, main topic, 1-10
  how to access, 3-4
  interface, 1-9
  local date exit, 4-26
  performance enhancements, 1-10
  performance versus transparency, 4-19
  performing distributed queries, 3-5
  SQL differences, 1-9
  tracing
    SQL statements, 3-8
  using, 3-1
  with other Oracle products, SQL*Plus, 1-6
GENERAL and DB2SQL linkage convention
gateway support, 4-5
GLOBAL_NAMES
  known restrictions, 2-4
globalization support
  Heterogeneous Services, B-5
GRAPHIC datatype, 4-23
graphic string operations
  unsupported, 4-23
GROUP BY clause
  SQL Set Clauses, 4-21
HAVING clause
SQL Set Clauses, 4-21
heterogeneous database integration, 1-10
Heterogeneous Services
  defining maximum number of open cursors, B-7
  optimizing data transfer, B-7
  setting global name, B-5
  specifying cache high water mark, B-5
  tuning internal data buffering, B-7
  tuning LONG data transfer, B-6
Heterogeneous Services (HS), see HS, 1-1
host
  performing character string operations on, 4-23
  relationship to gateway and Oracle database, 1-7
  variable, 4-21
HS (Heterogeneous Services)
architecture features, 1-10
HS_CALL_NAME initialization parameter, B-3
HS_DB_DOMAIN parameter
  known restrictions, 2-4
HS_DB_NAME initialization parameter, B-5
HS_DESCRIBE_CACHE_HWM initialization parameter, B-5
HS_FDS_CONNECT_INFO, B-9
HS_FDS_FETCH_ROWS parameter, B-10
HS_FDS_PACKAGE_COLLID parameter defined, B-14
HS_FDS_TRACE_LEVEL initialization parameter, B-11
  enabling agent tracing, B-2
HS_FDS_TRANSACTION_ISOLATION parameter, B-13
HS_FDS_TRANSACTION_LOG initialization parameter, B-11
HS_KEEP_REMOTE_COLUMN_SIZE initialization parameter, B-15
HS_LANGUAGE initialization parameter, B-5
HS_LONG_PIECE_TRANSFER_SIZE initialization parameter, B-6
HS_NLS_DATE_FORMAT
  four date patterns, 4-26
HS_OPEN_CURSORS initialization parameter, B-7
HS_RPC_FETCH_REBLOCKING initialization parameter, B-7
HS_RPC_FETCH_REBLOCKING parameter
  Oracle database support, 4-2
HS_RPC_FETCH_SIZE initialization parameter, B-7
HS_RPC_FETCH_SIZE parameter
  specified in the Gateway Initialization File, 1-11
  value determines array size, 4-2
HS_RPC_FETCH_SIZE parameter (continued)
I
IFILE initialization parameter, B-9
implementation, 1-8
implicit data conversion, 4-19
implicit protocol conversion, 1-4
Initialization parameter file
  customizing, B-1
input bind variables, 4-25
INSERT
  operation, 4-1
  SQL clause, 4-26
  statement
    passthrough SQL feature, 4-28
INSERT command
  supported by COPY, 3-8
Internet support, 1-3
INTERSECT, SQL set operators and clauses, 4-21
ISO standard, 1-4
isolation level,
  HS_FDS_TRANSACTION_ISOLATION, B-13
JOIN capability, 1-3
JOIN SQL statement, 4-1
K
known restrictions
  datatype limitations, 2-4
  date arithmetic, 2-5
  DD basic tables and views, 2-4
  GLOBAL_NAMES parameter, 2-4
  LONG datatype in SQL*Plus, 2-5
  null values and stored procedures, 2-4
  row length limitation, 2-5
  SUBSTR function post-processed, 2-4
languages
  access through the gateway, 1-5
  SQL*Plus, 1-6
link, also see Database Link, 4-9
literal
  character literals, 4-24
literal (continued)
  date, 4-24
  specific datatype, 4-21
  LONG RAW datatype, 4-23

M
  MINUS
    set operator, SQL Set Clauses, 4-21
    SQL set operators and clauses, 4-21
  Mobile Agents, 1-4

N
  Native Semantics
    gateway architecture, 1-10
    parameters, SQL Set Clauses, 4-21
    with SUBSTR function, known restrictions, 2-4
  non-character binary data, 4-23
  null
    rows, mapping the COUNT function, 4-27
    values
    mapping the COUNT function, 4-27
  numeric datatype
    zoned decimal column, 4-27

O
  o2pc.sql
    two-phase commit transactions, 3-6
  OPEN_LINKS parameter, 3-4
  operations
    DELETE, 4-1
    INSERT, 4-1
    SELECT, 4-1
    UPDATE, 4-1
  operators
    UNION ALL, SQL Set Clauses, 4-21
    UNION, SQL Set Clauses, 4-21
  option
    data dictionary views, 4-30
    date format string, 4-26
    FOR BIT DATA, 4-23
    Oracle database, 1-7
    read-only
      gateway configuration, 1-6
      replicating, 3-7
    SQL functions, 4-19
    SQL*Plus COPY command, 3-8
  ORA-02019 error, 5-1
  ORA-28500 error
    was ORA-09100, 5-2
  ORA1 Oracle instance, 4-2

Oracle
  mapped error codes, 5-3
  products compatibility, 1-9
  snapshots, 3-7

Oracle database
  accessing the gateway, 3-4
  architecture, 1-7
  copying data
    from DRDA server, 3-8
    to DRDA server, 3-7
  definition, 1-7
  relationship to host, 1-7
  services
    database triggers, 1-3
    distributed capabilities, 1-3
    distributed query optimization, 1-3
    extended database services, 1-3
    stored procedures, 1-3
    two-phase commit protection, 1-3
    stored procedure, defined, 4-2
    triggers, 3-7
    using, in application development, 4-1

Oracle Net
  and application development, 1-13
  and remote data access, 1-10
  and server coexistence, 1-7
  integrated with Oracle database, 1-5
  purpose, 1-8
  TNS connect descriptor specification, 3-3

ORARECOV user ID
  HS_FDS_RECOVERY_ACCOUNT, B-10

ORDER BY clause
  SQL Set Clauses, 4-21

P
  package
    collection id, HS_FDS_PACKAGE_COLLID, B-14
  packed decimal, 4-27
  parameter
    Native Semantics, SQL Set Clauses, 4-21
    setting up trace parameters, 5-3
  parameters
    DB_DOMAIN
      known restrictions, 2-4
    gateway initialization file
      HS_FDS_CAPABILITY, B-13
      HS_FDS_FETCH_ROWS, B-10
      HS_FDS_PACKAGE_COLLID, B-14
      HS_FDS_TRANSACTION_ISOLATION, B-13
    HS_DB_DOMAIN
      known restrictions, 2-4
    HS_NLS_DATE_FORMAT, 4-26
parameters (continued)
  HS_RPC_FETCH_REBLOCKING, 4-2
  HS_RPC_FETCH_SIZE, 1-11, 4-2
  OPEN_LINKS, 3-4
passthrough
  gateway features, 1-10
  native DB2 SQL, 1-5
  send SQL statement directly to DRDA server, 4-28
performance, 4-19
performance enhancements
  with fetch reblocking, 4-2
PL/SQL
  call, 4-4
  DRDA stored procedures, 4-3
  records, 4-5
  routine, 1-5
  standard Oracle, 1-5
  stored procedure, 4-2
post-processed SQL functions
  overview, 4-10
post-processing
  native semantics, 4-18
  SQL tracing in the gateway, 5-3
PREPARE TRANSACTION statement, 3-6
privileges
  data dictionary emulation, 4-30
procedure
  stored
    using DRDA server, 4-3
processing time, with GROUPBY, HAVING,
WHERE, 4-21
protocol
  commit confirm, 1-6
  converter, 1-4
  implicit protocol conversion, 1-4
  network, 3-5
  protocol-independent encryption, 1-4
  two-phase commit, 3-6
protocols
  TCP/IP
    gateway transparency, 1-2
    implicit protocol conversion, 1-4
remote (continued)
  data, 1-3
  data access, 1-10
  database
    creating database links, 3-3
    defining a path, 3-2
    errors detected by the Oracle database, 5-1
  DRDA database,
    HS_FDS_TRANSACTION_ISOLATION, B-13
  instance, and Oracle stored procedures, 4-2
  Oracle instance
    using DRDA server stored procedures with
    the gateway, 4-4
    using Oracle stored procedures with the
    gateway, 4-3
    procedure, 1-5
    table, 1-4
    user ID and password, 3-2
remote functions
  referenced in SQL statements, B-3
  REPLACE command, supported by COPY, 3-8
  replication, 3-7
  RESULT, 4-4
result sets
  columns in, 1-11
  retrieving result sets through passthrough, 1-10
REVISE_SALARY
  stored procedure, 4-4
S
  security
    Advanced Security, 1-4
    site autonomy, 1-6
SELECT and array size, 1-11
SELECT operation, 4-1
SELECT statement
  fetch reblocking, 4-2
  retrieving results sets, 4-29
SELECT WHERE
  SQL clause, 4-26
  semantics, 4-19
  session
    connection, 4-9
set operators
  compatibility, SQL Set Clauses, 4-21
  EXCEPT, SQL Set Clauses, 4-21
  INTERSECT, SQL Set Clauses, 4-21
  MINUS, SQL Set Clauses, 4-21
site autonomy, 1-6
snapshots
  known restrictions, SQL limitations, 2-6
  Oracle Snapshot feature, 3-7
queries, distributed, 3-5

RAW datatype
  performing character string operations, 4-23
  read-only support, 1-10
remote
  connections, 3-4
sort order
with ORDERBY, 4-21

SQL
ANSI standard, 1-4
clause compatibility, 4-21
clauses
DELETE, 4-26
INSERT, 4-26
SELECT WHERE, 4-26
UPDATE, 4-26

constructs
Oracle processing, 4-9
differences in the gateway, 1-9
errors mapped to Oracle error codes, 5-3
functions
SUBSTR, 4-19
functions and Native Semantics, 4-19
gateway transparency, 1-4
ISO standard, 1-4
native DB2, 1-5
passthrough, 4-28, 4-29
statements, 4-1
DB2, 4-29
issued through the gateway, 3-8
passing through gateway, 4-28
statements,
HS_FDS_TRANSACTION_ISOLATION, B-13
syntax, 4-27
tracing, not to be used in production environment, 3-8

SQL functions
column functions, 4-9
compatible, defined, 4-9
compensated, defined, 4-9
DB2/400, 4-16
DB2/OS390, 4-10
DB2/UDB, 4-13
post-processing, defined, 4-10
that can be disabled, 4-21
that can be enabled, 4-19
translated, defined, 4-9
with Native Semantics, 4-19

SQL tracing
in Oracle database, 5-3

SQL*Plus
COPY command, 3-8
moving data, 1-6

statements
CREATE DATABASE LINK, 3-2
DB2 CREATE TABLE, 1-5
DDL, 4-28, 4-29
DELETE, 4-26, 4-28
DROP DATABASE LINK, 3-3
INSERT, 4-28

statements (continued)
PREPARE TRANSACTION, 3-6
SELECT, 4-2, 4-29
SQL
DB2, 4-29
JOIN, 4-1
SELECT, 4-29
UPDATE, 4-26, 4-28

stored procedure
creating on DB2, 4-4
DB2, 4-5
native DB2, 1-5
Oracle and non-Oracle, 1-5
Oracle database
local instance, 4-2
PL/SQL, 4-2
remote instance, 4-2
using, 4-2
Oracle, description, 1-5
restriction, 2-4
using DRDA server, 4-3
stored procedures, 1-3
DB2, 4-3
REVISE_SALARY, 4-4
using with the gateway, 4-2

string index, with Native Semantics, 4-19
SUBSTR SQL function, 4-19
known restrictions, 2-4
with Native Semantics, known restrictions, 2-4

synonym
feature, 3-5
for location transparency, 4-3
how the gateway works, 1-9

T

table
create a table in DB2, 4-29
insert a row into a DB2 table, 4-29
TABLE_PRIVILEGES view, A-10
TCP/IP
facilities, 1-12
functions, 1-9
protocol
gateway transparency, 1-2
implicit protocol conversion, 1-4
support, 1-10
terminology defined, 1-7
TIME datatype, 4-24
time operations, 4-24
TIMESTAMP datatype, 4-24
tnsnames.ora
connect descriptor, 3-2
TO_DATE function, main topic, 4-26
tools and the gateway, 1-9
trace parameters
setting, 5-3
tracing
SQL statements, 3-8
trade-off, Native Semantics, 4-19
transparency
main topic, gateway transparency, 1-2
native semantics, 4-19
triggers for Oracle database, 3-7
two-phase commit
processing transactions, 3-6
protection, 1-3
unsupported statement, 3-6

U

UNION
 capability, 1-3
 operator, SQL Set Clauses, 4-21
 SQL set operators and clauses, 4-21
UNION ALL
 operator, SQL Set Clauses, 4-21
 SQL set operators and clauses, 4-21
UPDATE
 known restrictions, SQL limitations, 2-6
 operation, 4-1
 SQL clause, 4-26
 statement, 4-28
user privileges, 4-30
USER_CATALOG view, A-11
USER_COL_COMMENTS view, A-11
USER_CONS_COLUMNS view, A-12
USER_CONSTRAINTS view, A-11
USER_DB_LINKS data dictionary view, 3-4
USER_INDEXES view, A-12
USER_OBJECTS view, A-13
USER_SYNONYMS view, A-14
USER_TAB_COLUMNS view, A-16
USER_TAB_COMMENTS view, A-16
USER_TABLES view, A-14
USER_USERS view, A-17
USER_VIEWS view, A-17
USING clause, 3-2
Using the gateway, 3-1

V

VALUES clause
 functions not allowed by DB2, 4-26
 variable
 bind, SQL passthrough, 4-28
 input bind, 4-25
 view
catalog
 DB2/400, 4-30
 DB2/OS390, 4-30
 DB2/UDB, 4-30
data dictionary
 emulation on DRDA server, 4-30

W

WHERE clause
 known restrictions, SQL limitations, 2-6
 SQL Set Clauses, 4-21
WHERE CURRENT OF CURSOR clause
 known restrictions, SQL limitations, 2-6
wireless communication, 1-4

Z

zoned decimal operations, 4-27