Oracle® Database Provider for DRDA

User's Guide

Release 19c
E96456-01
January 2019

Oracle Database Provider for DRDA User's Guide, Release 19c

E96456-01

Primary Author: Tanmay Choudhury

Contributing Authors: Tulika Das

Contributors: Peter Castro, Charles Benet, Mark Jones, Roger Ford, Peter Wong

Contents

4 SQL Translation and Examples for Oracle Database Provider for DRDA

7 Datatype Support and Conversion in Oracle Database Provider for DRDA

12 Restrictions on Using Oracle Database Provider for DRDA

13 PL/SQL Packages Used by Oracle Database Provider for DRDA

14 SQL Statement Support in Oracle Database Provider for DRDA

List of Tables

Preface

Oracle Database Provider for DRDA User's Guide s describes how to migrate to Oracle Database while keeping DB2-based applications largely unchanged.

Audience

This document is intended for anyone who plans to migrate DB2- based applications to Oracle Database. Users should have knowledge of Oracle Database concepts and administration.

Related Documents

For more information, see the following documents in the Oracle Database documentation set:

- *Oracle® Database Concepts*
- *Oracle® Database Administrator's Guide*
- *Oracle® Database Migration Guide*
- *Oracle® Database SQL Language Reference*
- *Oracle® Database Reference*

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.

Conventions

The following text conventions are used in this document:

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

1

Introduction to Oracle Database Provider for DRDA

Consider some of the basic concepts of Oracle Database Provider for DRDA technology.

What is Oracle Database Provider for DRDA?

The Oracle Database Provider for DRDA is a network front-end that enables client programs to connect to Oracle Database using the Distributed Relational Database Architecture (DRDA) protocol.

Oracle Database Provider for DRDA implements a large subset of the full DRDA Version 4 specification, and several aspects of other DRDA server products (such as IBM's DB2) for compatibility. It is a database-independent protocol, and it provides only the functionality available from Oracle Database. Oracle Database Provider for DRDA product enables existing DB2 application customers to leverage their current investment in application technology while migrating from DB2 server.

Client programs or systems that use the DRDA protocol are called Application Requesters (ARs). Server programs or systems that provide DRDA protocol services, such as Oracle Database Provider for DRDA, are called Application Servers (AS).

Applications that are written to use the DRDA protocol, either as direct ARs or through an intermediate interface (such as embedded SQL), generally do not need to change their existing code to connect to Oracle Database through Oracle Database Provider for DRDA. Only minimal application configuration changes, such as retargeting, are necessary to successfully change the client application environment to use Oracle Database.

Release Information

Oracle Database Provider for DRDA is a network front-end that enables client programs to connect to Oracle Database using the Distributed Relational Database Architecture (DRDA) protocol. DRDA is a database-independent protocol. It provides only the functionality available from Oracle Database. Oracle Database Provider for DRDA enables existing DB2 application customers to leverage their current investment in application technology while migrating from DB2 server.

DB2 Client Applications

There are two general classes of DB2 applications:

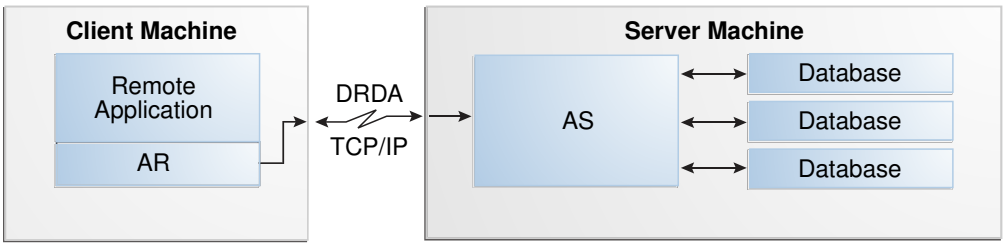
- Remote DB2 Applications
- Native DB2 Applications

Remote DB2 Applications

Remote applications use the DRDA data protocol to communicate with a target server database. The protocol's architecture is an example of a client/server model that includes the following, as illustrated in Figure 1-1:

- Client component, DRDA Application Requester (AR)
- Network substrate, such as a TCP/IP network or an SNA/APPC network
- Server component, Application Server (AS)

Figure 1-1 DRDA Connectivity Model



The application uses the AR to communicate with an AS, which in turn communicates with the database. In this configuration, applications are indirectly aware of the network because the AR connects to the network. The application does not require direct knowledge of the network connectivity.

Typically, DRDA AR implementations provide a directly callable API that may be coded by an application writer, such as ODBC. This API may also be invoked as part of a language pre-processor that translates source code with embedded SQL statements into equivalent embedded API calls. This is similar in concept to Oracle's OCI API and Oracle's Pro*C preprocessor products. In both cases, the application is agnostic with respect to the actual database connectivity; only specific API calls attach the application to the database.

Within a network, client/server architecture has cross-platform interoperability: the client and the server may run on any supported computer platform. For example, IBM makes DB2 Database server product available on AS/400, z/OS, VM, VSE, Linux, several Unix platforms, as well as Microsoft Windows. IBM also makes clients, such as IBM's DB2 Connect product, available on several platforms. This arrangement enables the client to communicate with several servers and to be easily redirected to a different server, which may be on the same or different remote host.

Examples of remote applications include:

- ODBC-based applications
- Java/JDBC-based applications
- DB2 Database, used for remote database-to-database connections
- DB2 Connect, used to redirect native applications
- Custom applications that use one of several available AR implementations

ODBC- and JDBC-enabled applications may be retargeted to use Oracle Database Provider for DRDA with little or no change to the application itself.

IBM's DB2 products have native support for DRDA, where DB2 may be a requester, a server, or both. This book only discusses the scenarios where DB2 is a requester to the Oracle Server.

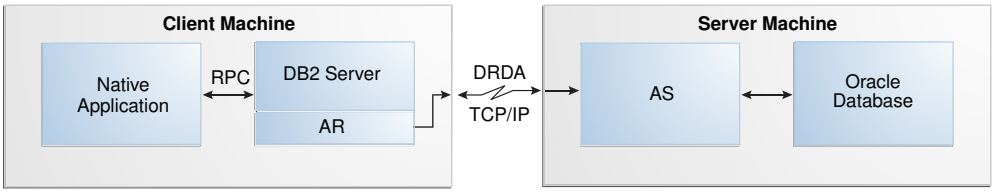
Related Topics

- *Oracle Call Interface Programmer's Guide*
- *Pro*C/C++ Programmer's Guide*

Native DB2 Applications

Native Applications are supported by Oracle Database Provider for DRDA, but they require an existing DB2 database server to redirect the network. This is because native DB2 applications are more tightly intertwined with the DB2 server. This class of application communicates directly with a specific DB2 server using a local and proprietary API. While such applications cannot directly connect to other databases, they can use a remote node connectivity mechanism to connect indirectly to a remote database. This is illustrated in Figure 1-2.

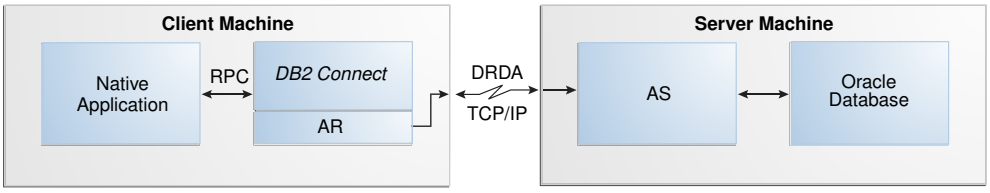
Figure 1-2 Native Application Remote Connectivity Model



This is not an ideal approach, because using a full DB2 server merely to provide remote access to native applications is very cost-prohibitive. This is both because of a licensing model's "per processor" structure, and the disk and memory footprint.

A more attractive alternative to a full multiprocessor DB2 server is to use a local application, such as DB2 Connect, to provide remote connectivity. In such cases, the applications' access can be converted to network connectivity through DRDA, as illustrated in Figure 1-3.

Figure 1-3 DB2 Connect Replacement of DB2 Server Connectivity Model



In some cases, it is not possible to replace the DB2 database server with an alternative native application enabler. Such applications include the following:

- CICS DB2-connected applications on z/OS
- DB2/400 native applications

- DB2 for z/OS native applications
- DB2 for Linux, Unix, and Windows native applications

In these situations, the application can connect, by proxy, through the local DB2 database server. While this is not an ideal approach, it reduces the investment in DB2 server products. If an application does not use the DB2 database product, the number of DB2 servers may be reduced to the DB2 instances that are necessary as application proxies.

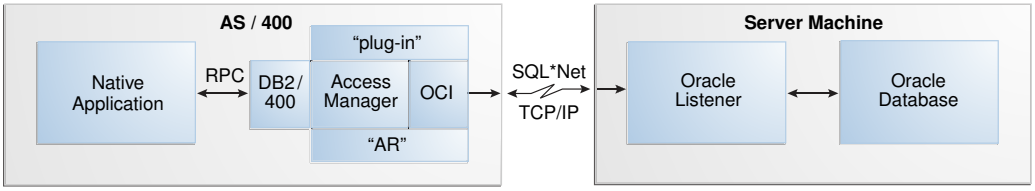
Usage Scenarios for Oracle Database Provider for DRDA

Sections Remote DB2 Applications and Native DB2 Applications describe some possible usage scenarios for an Oracle Database Provider for DRDA. Obviously, both remote and native applications may be retargeted to use Oracle Database through Oracle Database Provider for DRDA.

Because Oracle Database Provider for DRDA is a network solution, the network infrastructure should have sufficient excess capacity to accommodate increased load when retargeting native applications. In cases of remote applications the data flow between the client and the server does not change significantly.

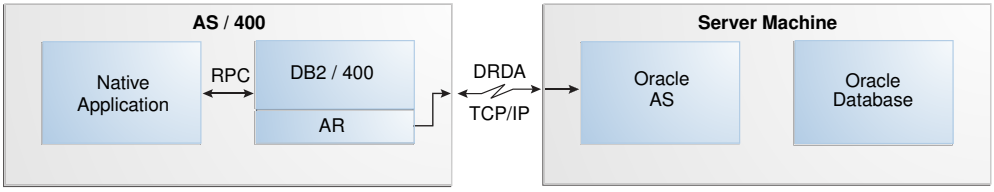
While most scenarios revolve around a standard AR, usually supplied by IBM, there is a case where the Application Server replaces Oracle Access Manager for AS/400, which has been discontinued. Access Manager is a client-side product that enables native DB2/400 applications to connect to Oracle Database as if it were a remote DB2 database. The Access Manager runs on the AS/400 as a DB2/400 API plug-in and uses an OCI method to connect to Oracle Database, as illustrated in Figure 1-4.

Figure 1-4 Access Manager Plug-in Architecture Connectivity Model



DB2/400's plug-in interface API behaves like DRDA, and appears to the system as an application requester that uses OCI and SQL*Net internally to connect to an Oracle Database. Customers who must connect a native application from the AS/400 to an Oracle Database will find Oracle Database Provider for DRDA a more cost-effective solution. This approach is illustrated in Figure 1-5.

Figure 1-5 DB2/400 Native DRDA Usage Connectivity Model



The example of the Access Manager is just one such scenario where a native application can be retargeted to an Oracle Database instance through the Application Server.

All of the scenarios discussed here can use Oracle Database Provider for DRDA to connect to Oracle Database.

Related Topics

- Remote DB2 Applications

Related Topics

- Native DB2 Applications

2

Architecture of Oracle Database Provider for DRDA

Consider the architecture of Oracle Database Provider for DRDA.

For more information about DRDA in DB2, see *DB2 Version 9.1 for z/OS Information Center* at http://www.ibm.com. More specifically, see http://www.redbooks.ibm.com/redbooks/pdfs/sg246952.pdf.

Protocol Considerations

DRDA is a data protocol with some similarities to Oracle's SQL*Net data protocol. While DRDA is designed to move relational data between a client and a server, it lacks the more robust management and routing controls of SQL*Net. The primary difference between DRDA and SQL*Net is the language of the protocol itself. DRDA and SQL*Net are not compatible, so it is not possible to use a SQL*Net client to connect to a DRDA server, or vice versa.

The terminology used with DRDA is also similar to SQL*Net, and general concepts translate to conventional Oracle definitions, as demonstrated by the following examples:

- An ***application requester*** (AR) is an interface that client programs use to create and send SQL-based requests to an application server.
- An ***application server*** (AS) is a server and database-side program that accepts such requests on behalf of the client, executes database operations, and returns resulting data back to the client.

Two-Phase Commit and Transaction Recovery

DRDA and DB2 implement two command sets that enable commit and rollback of transactions. They ensure that data integrity is maintained during updates of a transaction, and that these updates may be recovered if either the connection or applications fail at the time that the transaction is being committed. DRDA supports commands that implement both **SingleSite** and **TwoPhase** commit protocols. At a minimum, the AS must support SingleSite commitment.

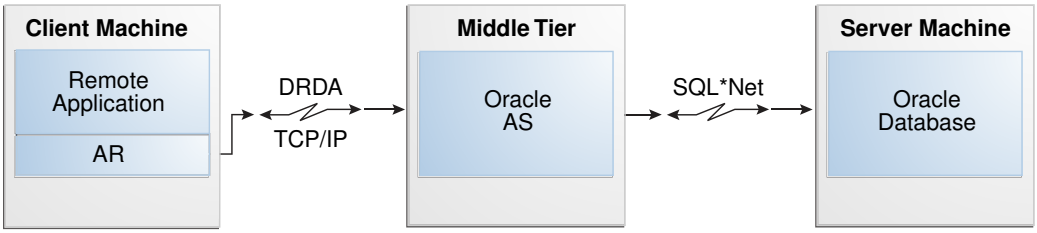
- Single Site commit protocol consists of a simple operation that has no ability to coordinate between nodes that may be involved in a distributed transaction. It is the basic mechanism for committing data, and it is used by most common applications.
- Two Phase commit protocol enables the coordination of multiple transactions, either on the same node or on separately networked nodes.

Autonomy of Service

The Application Server is external to the Oracle Database server. Because of this, the application has a wide range of location options.

In a typical configuration, the AS runs on the same machine as the Oracle Database, as described in the DRDA Connectivity Model, under section Remote DB2 Applications. Because the AS is not tightly integrated into the Oracle Database, it may be installed into its own Oracle Home, or on a machine separate from either the client or the database, as illustrated in Figure 2-1. This **middle tier configuration** enables the separation of service resources. It also allows for better scaling of service because separate machine resources can be dedicated to both AS and Database.

Figure 2-1 AS Middle Tier Configuration



Additionally, the separation of the AS from the Client and Server tiers provides an extra layer of security and reliability. If the AS crashes, the Oracle Database instance is not impacted. Database integrity is maintained, and Oracle Database recovers the state of the transaction.

Related Topics

- Remote DB2 Applications

Packages

The resources associated with DRDA application are known as **packages**. More specifically, the application requester utilizes a package as a reference to what the application does: the package is where the statements are stored. The application refers to the statement through a section number. There are two general classes of application statements: static and dynamic.

- **Static statements** contain hard-coded SQL, statements where the SQL text does not change during the run of the application. They are very quick to execute, and are often optimized prior to run-time to achieve high performance. Because it is predefined, static SQL has a shorter execution time at first invocation.
- **Dynamic statements** are primarily empty placeholders, and are sometimes called **generic cursors**. They have no SQL text before run-time, and the application constructs the actual SQL statements it needs during operation and optimizes them at runtime. After the first invocation, processed dynamic SQL statements are typically cached, so subsequent execution time of the same statement is comparable to static SQL statements of similar complexity.

The packages are constructed through proprietary tools. For example, in a DB2 application environment, a developer often writes an application that contains

embedded SQL statements. The application source is processed by the SQL PreCompiler, which is analogous to Oracle's Pro*C precompiler. The output is typically post-processed into a source module, along with an on-disk resource form of the statements used in the source program. In DB2 terminology, this creates a Database Request Module, or DBRM. Most implementations that create this file store it externally in a proprietary format.

The contents of the DBRM must be either loaded into the remote database or otherwise made available to the AS at the time of execution. However, loading data that is in proprietary format has many challenges. DRDA addresses this by providing a set of Command Requests to remotely upload the resource definition into a target AS. Most AR implementations provide an option or tool to upload the resource, either before or during the application's SQL session. This process is called **binding a package**.

After the resource definition (DBRM) is bound as a package to the remote database, the AS may load it in advance for better performance.

SQL Dialect

While Oracle is partially ANSI SQL compliant, as are most SQL-based database systems today, there are some exceptions. Database vendors implemented the ANSI SQL standard differently; this resulted in SQL 'dialects' that present some challenges during statement execution. Because the original target database used with DRDA is DB2, the applications that are discussed here use the DB2-specific dialect of SQL.

Much of Data Manipulation Language (DML) and some Data Definition Language (DDL) has been standardized in ANSI SQL for commonly used objects such as tables, views, indexes, simple procedures, or function definitions. However, each database vendor will still have its own set of product-specific extensions to both DDL and DML. The DRDA protocol treats SQL statements as database-specific entities that the database must handle; it indicates to Oracle Database that its SQL is in a DB2 dialect, and that it must have a translation service to handle it. This translation is supplied by the SQL Translation Framework feature, fully described in Oracle Database SQL Translation and Migration Guide.

3

Installation and Configuration of Oracle Database Provider for DRDA

Consider installation, configuration, and administration of Oracle Database Provider for DRDA.

About Installing Oracle Database Provider for DRDA

Installation involves starting the Oracle Universal Installer, entering the Oracle home path, and selecting "Oracle Database Provider for DRDA" product to be installed. The installation will ask for several initial configuration items in an Interview panel for the product.

Note that the following procedure creates the `drdaas.ora` configuration file in the directory `$ORACLE_HOME/drdaas/admin/`.

Related Topics

- Configuration File: drdaas.ora

installing Oracle Database Provider for DRDA

1. Start Oracle Universal Installer.
2. Enter the path of an existing *ORACLE_HOME*, or a new path for a stand-alone installation.
3. [Optional] Enter the *ORACLE_HOME* name.
4. Select Oracle Database Provider for DRDA for installation.
5. In the **Interview** panel, enter the following information:
   - Oracle Database Provider for DRDA listener host name and/or IP address

     Specify host name or IP address of the network interface where Oracle Database Provider for DRDA is the listener.

     Default is an empty string.
   - Oracle Database Provider for DRDA listener port number

     Specify the port number of Oracle Database Provider for DRDA Listener.

     Default is `1446`.
   - Oracle Database Provider for DRDA RDB map name

     Specify the external relational database name that the client applications use as a location qualifier.

     Default is `DRDAAS`.
   - Oracle Database connection descriptor

Specify the connection descriptor to use to connect to the Oracle Database. Valid values are:

- `TNS(tns_name_sentry)`, such as `TNS(orcl)`
- `oracle_sid` or `ORACLE_SID`

  This accesses the local Oracle Database before Oracle Database Provider for DRDA starts, based on the setting of the environment variable `$ORACLE_SID`.
- TNS-Descriptor

  For example:

  ```
  DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=drdaas))
  (CONNECT_DATA=(SERVICE_NAME=drdaas.us.oracle.com)));
  ```

  Default is `oracle_sid`.

6. Select **Finish**.
7. At the command prompt, create a diagnostic directory:

   ```
   % mkdir -p $ORACLE_HOME/log/diag/dps
   ```

Configuring Oracle Database Provider for DRDA

To configure Oracle Database Provider for DRDA, you must update the `drdaas.ora` configuration file with any necessary information.

Additionally, you must install the objects that depend on Oracle Database Provider for DRDA product in all Oracle Database instances that use Oracle Database Provider for DRDA.

Updating the drdaas.ora Configuration File

Typically, no additional parameters are needed beyond those specified during installation.

In more advanced scenarios, it may be necessary to specify more than one `DATA_PORT` parameter or configure more than one `RDB_MAP` entry. Still more complex installations may require multiple Oracle Database Provider for DRDA instances.

For a list of parameters and their options, refer to the section Configuration Parameters.

Related Topics

- Configuration Parameters

DATA_PORT Considerations

Additional `DATA_PORT` entries may be specified with different host name or IP addresses, and unallocated network port numbers. This is called a *Multiplexed Instance configuration*.

RDB_MAP Considerations

Additional `RDB_MAP` entries may be specified to add other map entries for converting between third-party relational databases and Oracle.

> **Note:**
>
> For some DRDA clients, such as the IBM DB2 Database for z/OS, the user must configure only one `RDB_MAP` entry for each Oracle Database Provider for DRDA instance. This is called a *Dedicated Instance configuration*.

Oracle Database Provider for DRDA Instance Considerations

It may be necessary to define more than one Oracle Database Provider for DRDA Instance to accommodate an environment that contains both types of DRDA clients. Therefore, Oracle Database Provider for DRDA product supports the definition of multiple instance configurations in the same `drdaas.ora` configuration file.

Installing Database Objects

There are two sets of database objects that must be installed: Global Objects and Per-User Objects.

About Global Objects

Each instance of Oracle Database used in an Oracle Database Provider for DRDA configuration must install Oracle Database Provider for DRDA specific objects. This involves the following procedures:

- Creating a `SYSIBM` tablespace
- Installing Oracle Database Provider for DRDA catalogs
- Installing DB2 SQL translator
- Designate Oracle Database Provider for DRDA administrative role

Related Topics

- Security and Storage Considerations for Oracle Database Provider for DRDA

Creating a SYSIBM tablespace

1. Connect to the database; this example uses `SYSDBA` privileges, but they are not necessary.

   ```
   > connect SYS as SYSDBA
   ```

2. Create the tablespace `SYSIBM`.

   ```
   > create tablespace SYSIBM datafile 'sysibm01.dbf' size 70M reuse extent
   management local segment space management auto online;
   ```

   This creates tablespace `SYSIBM` in the directory specified by the parameter `DB_CREATE_FILE_DEST`.

Related Topics

- SYSIBM Tablespace

Installing Oracle Database Provider for DRDA Catalogs

1. Change directory to $*ORACLE_HOME*/rdbms/admin

   ```
   $ cd $ORACLE_HOME/rdbms/admin
   ```

2. Connect to the database with SYSDBA privileges.

   ```
   > connect SYS as SYSDBA
   ```

3. Invoke the following SQL script:

   ```
   > @catdrdaas.sql
   ```

4. If using Oracle Database Release 12c, invoke the following SQL script:

   ```
   > @prvtdpsadzoscat.plb
   ```

   If using Oracle Database Release 11.2, invoke the following SQL script:

   ```
   > @prvtdpsadzoscat11.plb
   ```

Related Topics

- catdrdaas.sql

Installing DB2 SQL translator

The DB2 SQL Translator is available starting from Oracle Database 12*c*.

> **Note:**
>
> The SQL Translator is an optional feature that you can use, when the SQL used within the application consists of a lot of DB2 specific syntax.

1. Change directory to $*ORACLE_HOME*/drdaas/admin

   ```
   $ cd $ORACLE_HOME/drdaas/admin
   ```

2. Connect to the database with SYSDBA privileges.

   ```
   > connect SYS as SYSDBA
   ```

3. Invoke the following SQL script, and answer its prompts:

   ```
   > @drdasqtt_translator_setup.sql
   ```

   The script `drdasqtt_translator_setup.sql` is demonstrated in the section Code for SQL Translation Interface Package and a Translation Profile.

Related Topics

- Creating a SQL Translation Interface Package and Translation Profile

Designating Oracle Database Provider for DRDA Administrative Role

1. Designate one or more user IDs to be an Oracle Database Provider for DRDA administrator. This role may set the access authority for DRDA packages and associated DRDA package attributes. Oracle user `SYSTEM` may be used, but the privilege may be granted to any user who performs the functions of DRDA administrator.
2. Grant initial DRDA package binding authority. Invoke the following SQL script and answer the prompts:

```
> @drdapkg_db2.sql

SQL> Prompt Enter the OracleID under which the initial package BINDs will be made
SQL> Use quotes (') if needed.
SQL> Accept OracleID
DRDAUSR

SQL> Enter default collection ID for package binding (usually NULLID)
SQL> Use quotes (') if needed.
SQL> Accept DefaultCollection
NULLID
```

Related Topics

- DBMS_DRDAAS_ADMIN Package

SQL Translation Profile

To facilitate correct interpretation of DRDA-based application SQL from it's native DB2 dialect to Oracle, the user must create a SQL Translation Profile.

SQL Translation Profiles are managed on a per-user basis. In contrast, DRDA packages are managed on the application basis. As a result, only one SQL translation profile name may be associated with a specific DRDA package. The same SQL Translation Profile may be also associated with many packages; for consistency, the same SQL Translation Profile should be associated with each defined package.

To create an additional translation profile, DRDA users must request a profile name from the DRDA administrator and then invoke the `drdasqtt_translator_setup.sql` script, in `$ORACLE_HOME/drdaas/admin/` directory. The section ‘Create a SQL Translation Interface Package and Translation Profile demonstrates how to create a SQL Translation Profile for profile name `DB2ZOS`; this code creates the template of a translation profile.

> **Note:**
>
> SQL Translation Profile is an optional feature, and can be used only when the SQL Translation Feature is configured.

Related Topics

- Creating a SQL Translation Interface Package and Translation Profile

Prerequisites for Creating a SQL Translation Profile

This feature is available to users of Oracle Database 12*c* or higher.

The user must have `DRDAAS_USER_ROLE`, as described in the section Authorizing Oracle Database Provider for DRDA.

A DB2 SQL Translator must be loaded into the database. The user may create a translation profile using this translator.

Creating a SQL Translation Interface Package and Translation Profile

Example 3-1 creates a SQL Translator Interface Package `SYSIBM.DBTooIntPkg`, and a SQL Translation Profile `TRANS_ADMIN.MyDBTooTransProfile`. It assumes that the third-party SQL translator is in JAVA, and that it appears entirely within an object `ThirdPartyDB2Translator.class` in the `rdbms/drdaas/jlib` directory.

Additional translations may be added, changed, or removed as needed. Please refer to the *Oracle® Database SQL Translation and Migration Guide* for details.

Example 3-1 Code for SQL Translation Interface Package and a Translation Profile

The following two lines describe the signatures of the two translator methods within the third-party object:

```
ThirdPartyTranslator.translateSQL(oracle.sql.CLOB,oracle.sql.CLOB[])
ThirdPartyTranslator.translateError(int,int[],java.lang.String[])
```

These signatures determine the method Oracle calls to translate both SQL text and Oracle Error codes. The method `translateSQL()`has two arguments: a `CLOB` for the original SQL text, and a `CLOB` for the CLOB output from the SQL translator. The second method may be ignored.

```
connect / as sysdba
@$ORACLE_HOME/drdaas/admin/drdasqtt_translator_setup.sql

Enter schema in which the SQL Translator Interface Package will be created as well
as into which the third-party SQL translator will be loaded (usually SYSIBM).
SQL Translator Interface Package Schema:SYSIBM

Enter unqualified name of the SQL Translator Interface Package
SQL Translator Interface Package Name:DBTooIntPkg

Enter schema in which the Translation Profile will be created:
Translation Profile Schema:TRANS_ADMIN

Enter the unqualified name of the translation profile:
Translation Profile Name:MyDBTooTransProfile

Enter the "language" type of the translator: C, java, etc
Translator Language:JAVA

Enter the path names of the third-party SQL Translator objects;
(All objects must be located under the "rdbms/" directory,
for example: "rdbms/drdaas/jlib/objecta.jar").

Enter all path qualified objects, one per prompt, up to 10.
```

```
Enter "" for all remaining object prompts.
SQL Translator object#1: rdbms/drdaas/jlib/ThirdPartyDB2Translator.class
SQL Translator object#2: ""
...
SQL Translator object#10: ""

Enter the signature for the entry for 'translateSQL' in one of the
previously entered SQL Translator objects:
Entry for
 translateSQL:ThirdPartyTranslator.translateSQL(oracle.sql.CLOB,oracle.sql.CLOB[])

Enter the signature for the entry for 'translateError' in one of the
previously entered SQL Translator objects callout for
 translateError:ThirdPartyTranslator.translateError(int,int[],java.lang.String[])
```

Configuration File: drdaas.ora

The file `drdaas.ora` defines the instances of Oracle Database Provider for DRDA. This file is composed of initialization parameters that define the instances of the Application Server.

The file `drdaas.ora` may be customized. However, it may also be created at installation time from questions posed by the Installer and from user input.

The `drdaas.ora` configuration file must be located in the Oracle Home, under the product administration directory.

The default location is: `$ORACLE_HOME/drdaas/admin`.

Note that parameters that are qualified by the instance name apply only to that specific instance. Parameters that are not qualified by an instance name apply to all instances specified in the file.

Example 3-2 Sample configuration file, drdaas.ora

```
# Example pre-configured instance named "drdaas"
# defines a single port and an rdb map that uses
# the local database instance accessed through
# the ORACLE_SID environmental variable.
drdaas.DATA_PORT = 10.0.0.1:1446
drdaas.RDB_MAP = RDB(DB2DS4M)->ORACLE_SID
#
# Example instance using a single port and a single rdb map
drdasingle.DATA_PORT = 10.0.0.1:1546
drdasingle.RDB_MAP = RDB(DB2DSN1)->TNS(ora101)
#
# Example instance using multiple rdb mappings
drdamulti.DATA_PORT = 10.0.0.1:2446
drdamulti.RDB_MAP = RDB(DB2DSN1)->TNS(ora101)
drdamulti.RDB_MAP = RDB(DB2DSN2)->TNS(ora102)
drdamulti.RDB_MAP = RDB(DB2DSN3)->TNS(ora103)
#
# global section affects all instances unless overridden
PROTOPROC_TRACE="ALL ERROR"
```

Authorizing Oracle Database Provider for DRDA

Oracle users must have the appropriate Oracle Database Provider for DRDA role in order to access Oracle Database Provider for DRDA catalogs and specific DRDA packages.

Related Topics

- About Global Objects

Administrator Role

Users who must perform administrative functions must have the `DRDAAS_ADMIN_ROLE` role. This enables privilege grants on a specific DRDA package, and assigning package attributes (SQL translation profile name).

For installations that do not have the default role `ALL`, have several default roles for users, such as `CONNECT` or `RESOURCE`, and add the add `DRDAAS_ADMIN_ROLE` role to the default list.

The `DRDAAS_ADMIN_ROLE` role is not meant for users who must use the DRDA packages. They should have the `DRDAAS_USER_ROLE` assignment, instead.

Administration is mainly concerned with granting and revoking of privilege to users, setting attributes on packages, and dropping packages.

Users who create packages, or are designated as owners of a package, have implicit authority over that package and may grant access to others. For example, the package owner may grant `RUN` privileges to any number of users. An owner may also set package attributes and drop the package.

However, in order to bind a package initially, a user must have `BIND` privilege, either for any package in a collection, or specifically for that package. Only users who have the `DRDAAS_ADMIN_ROLE` role may grant authorization to users for packages that are not already bound, or are not created or owned by that user. For information on how to grant access to a package and how to set package attributes, refer DRDA Package Authorization.

Related Topics

- Granting DRDAAS_ADMIN_ROLE
- Adding DRDAA_ADMIN_ROLE
- Adding DRDAAS_USER_ROLE
- DRDA Package Authorization

Granting DRDAAS_ADMIN_ROLE

Example 3-3 Granting the DRDAAS_ADMIN_ROLE

```
connect sys as sysdba
grant DRDAAS_ADMIN_ROLE to DRDAADMIN;
```

Adding DRDAA_ADMIN_ROLE

Example 3-4 Adding DRDAAS_ADMIN_ROLE to Default Values

```
alter user DRDAADMIN default role CONNECT, DRDAAS_ADMIN_ROLE;
```

Dropping ORACLE.MYPACKAGE by Administrator

This function should be performed by a user with DRDA administrator role.

Example 3-5 Dropping package ORACLE.MYPACKAGE, as Administrator

```
connect DRDAADM/password
execute DBMS_DRDAAS_ADMIN.DROP_PACKAGE('ORACLE','MYPACKAGE);
commit;
```

Dropping ORACLE.MYPACKAGE by User

This function should be performed by a user with DRDA user role. This operation fails if user DRDAUSR2 does not own package ORACLE.MYPACKAGE, if the user is not the creator of this package, or if the user has no DROP privilege for this package.

Example 3-6 Dropping package ORACLE.MYPACKAGE, as User

```
connect DRDAUSR2/password
execute DBMS_DRDAAS.DROP_PACKAGE('ORACLE','MYPACKAGE);
commit;
```

Maintaining DRDA Packages

Another primary responsibility of an administrator is to clean old or unused packages from the system. A list of all packages may be found by querying table ALL_DRDAASPACKAGE:

```
SELECT * from ALL_DRDAASPACKAGE;
```

DRDA Package Authorization

The DRDA administrator must perform these functions before supplying the user with the DRDA package name and (optionally) the SQL translation profile name.

Only a DRDA administrator may grant access to specific DRDA packages.

Refer the section on Granting and Revoking a User's Package Privileges to understand how to grant the BIND, DROP and EXECUTE privileges to user DRDAUSR*x* for package ORACLE.MYPACKAGE.

The DRDA Administrator may also designate a SQL translation profile name to associate with the DRDA package.

Refer the section on Setting and Deleting Translation Profile Name for a DRDA Package to understand how to set the profile name to DB2ZOS.

Related Topics

- Managing a User's Package Privileges

Managing a User's Package Privileges

Example 3-7 Granting and Revoking a User's Package Privileges

```
connect DRDAADM/password

Rem Grant BIND on any package in collection ORACLE to DRDAUSR
execute DBMS_DRDAAS_ADMIN.GRANT_PRIVILEGE(
     DBMS_DRDAAS_ADMIN.BIND_PRIVILEGE, 'ORACLE', '*', 'DRDAUSR');

Rem Grant BIND on package ORACLE.MYPACKAGE to user DRDAUSR2
execute DBMS_DRDAAS_ADMIN.GRANT_PRIVILEGE(
     DBMS_DRDAAS_ADMIN.BIND_PRIVILEGE, 'ORACLE', 'MYPACKAGE', 'DRDAUSR2');

Rem Grant EXECUTE on package ORACLE.MYPACKAGE to PUBLIC
execute DBMS_DRDAAS_ADMIN.GRANT_PRIVILEGE(
     DBMS_DRDAAS_ADMIN.EXECUTE_PRIVILEGE, 'ORACLE', 'MYPACKAGE', 'PUBLIC');

Rem Grant SET on package ORACLE.MYPACKAGE to user DRDAUSR3
execute DBMS_DRDAAS_ADMIN.GRANT_PRIVILEGE(
     DBMS_DRDAAS_ADMIN.SET_PRIVILEGE, 'ORACLE', 'MYPACKAGE', 'DRDAUSR3');

Rem Grant DROP on any package in collection ORACLE to user DRDAUSR3
execute DBMS_DRDAAS_ADMIN.GRANT_PRIVILEGE(
     DBMS_DRDAAS_ADMIN.DROP_PRIVILEGE, 'ORACLE', '*', 'DRDAUSR3');

Rem Grant ALL on any package in collection NULLID to user DRDAUSR3
execute DBMS_DRDAAS_ADMIN.GRANT_PRIVILEGE(
     DBMS_DRDAAS_ADMIN.ALL_PRIVILEGE, 'NULLID', '*', 'DRDAUSR3');

Rem Revoke BIND on package ORACLE.NOTYOURPKG from user DRDAUSR3
execute DBMS_DRDAAS_ADMIN.REVOKE_PRIVILEGE(
     DBMS_DRDAAS_ADMIN.BIND_PRIVILEGE, 'ORACLE', 'NOTYOURPKG', 'DRDAUSR3');

Rem Revoke ALL on any package in Collection OTHER from user DRDAUSR4
execute DBMS_DRDAAS_ADMIN.REVOKE_PRIVILEGE(
     DBMS_DRDAAS_ADMIN.ALL_PRIVILEGE, 'OTHER', '*', 'DRDAUSR4');
```

Managing DRDA Package Translation Profile

Example 3-8 Setting and Deleting Translation Profile Name for a DRDA Package

```
connect DRDAADM/password

Rem Set the DB2ZOS profile name for "any" package in collection ORACLE
DBMS_DRDAAS_ADMIN.SET_PROFILE( 'ORACLE', '*', 'DB2ZOS' );

Rem Set the MYDB2ZOS profile name for package ORACLE.MYPACKAGE
DBMS_DRDAAS_ADMIN.SET_PROFILE( 'ORACLE', 'MYPACKAGE', 'MYDB2ZOS' );

Rem Deleting the profile name for package ORACLE.MYPACKAGE
execute DBMS_DRDAAS_ADMIN.SET_PROFILE( NULL, 'ORACLE', 'MYPACKAGE' );

Rem Deleting the profile name for "any" package in collection ORACLE
execute DBMS_DRDAAS_ADMIN.SET_PROFILE( NULL, 'ORACLE', '*' );
```

User Role

Each user who accesses the database through DRDA must have the `DRDAAS_USER_ROLE` user role as a default.

Users commonly have the default role `ALL`, which immediately enables all granted roles. If a user has explicit default roles, they must also have the `DRDAAS_USER_ROLE` role, as described in the section Adding DRDAAS_USER_ROLE to Default Values.

Note that failing to specify the complete list of default roles may prevent the user from connecting to the database, or form being able to address certain resources implicitly.

Refer to *Oracle® Database SQL Language Reference* and *Oracle Database Security Guide*.

Related Topics

- Adding DRDAAS_USER_ROLE

Granting the DRDAAS_USER_ROLE

Example 3-9 Granting the DRDAAS_USER_ROLE

```
connect sys as sysdba
grant DRDAAS_USER_ROLE to DRDAUSR;
```

Adding DRDAAS_USER_ROLE

Example 3-10 Adding DRDAAS_USER_ROLE to Default Values

```
alter user DRDAUSR default role CONNECT, RESOURCE, DRDAAS_USER_ROLE;
```

Uninstalling Oracle Database Provider for DRDA

Full uninstall of Oracle Database Provider for DRDA involves the removal of the Database objects and uninstall of Oracle Database Provider for DRDA software. The product-dependent objects are removed, while the customer data remains intact.

Removing the Database Objects

The following steps remove all Oracle Database Provider for DRDA objects from the database, and drop the user-created tablespace.

To remove Database objects:

1. Change directory to $*ORACLE_HOME*`/rdbms/admin`.

   ```
   > cd $ORACLE_HOME/rdbms/admin
   ```

2. Connect to the database using the `SYSDBA` option.

   ```
   connect / as sysdba
   ```

3. Run the removal script.

   ```
   catnodrdaas.sql
   ```

4. Drop the user-created tablespace `sysibm` and its contents.

```
> drop tablespace sysibm;
```

See *Oracle® Database SQL Language Reference* for `DROP TABLESPACE` options.

Uninstalling Oracle Database Provider for DRDA software

To uninstall Oracle Database Provider for DRDA software, you must use the Oracle Universal Installer with the `-deinstall` option. You may choose specify path to Oracle Home using the `-home` option, or choose Oracle Home when Oracle Universal Installer is running.See further instructions on uninstalling Oracle software in *Oracle® Database Installation Guide*.

To uninstall Oracle Database Provider for DRDA software:

1. In Oracle Universal Installer, click **Installed Products**.
2. Select the desired *`ORACLE_HOME`*.

   Either select the whole *`ORACLE_HOME`*, or open *`ORACLE_HOME`* navigation tree and select Oracle Database Provider for DRDA product software.
3. Click **Remove**.

Configuration Parameters

Oracle Database Provider for DRDA uses several parameters to configure its environment; they are specified in the configuration file.

DATA_PORT

This designates the DRDA data port used by this instance, and is represented by an Internet Address and Port Number.

Default Value

There is no default port number; an explicit port number must be specified. Oracle recommends using `1446`.

Allowable Values

A valid, unallocated TCP/IP network port number, optionally prefixed with a specific host name or IP address associated with a defined network interface on the local machine.

Syntax

`DATA_PORT = {`*`host_name`*`|`*`ip_address`*`:}`*`number`*

Usage Example

```
DATA_PORT = 10.0.0.1:1446
```

RDB_MAP

This `string` parameter maps relational database names, as passed in the DRDA `ACCRDB` command object, to Oracle TNS entries, or to the locally addressable Oracle instance. This parameter may contain several occurrences of a map entry.

This has no default value.

A list of optional values includes the following:

- `tns_name_entry` corresponds to a TNS entry in the local `tnsnames.ora` configuration file.
- `tns_entry` is a fully-formed TNS descriptor string. It may be used instead of a TNS name entry.
- `oracle_sid` uses the `$ORACLE_SID` environment variable value that is set prior to starting an Oracle Database Provider for DRDA instance.

Note that the use of one or more occurrences of `RDB_MAP` determines a mode of compatibility with older application requesters. The default is a single, dedicated definition that connects to a single Oracle Database instance based on the `ORACLE_SID` environmental variable.

Allowable Values

A valid, unallocated TCP/IP network port number, optionally prefixed with a specific host name or IP address associated with a defined network interface on the local machine.

Syntax

```
RDB_MAP = RDB(rdb_name)->TNS(tns_name_entry)
RDB_MAP = RDB(rdb_name)->ORACLE_SID
RDB_MAP = RDB(rdb_name)->"tns_entry"
```

Usage Examples

```
RDB_MAP = RDB(DB2DSN1)->ORACLE_SID
RDB_MAP = RDB(DB2DSN2)->TNS(ora101)
RDB_MAP = RDB(DB2DSN3)->"(ADDRESS=(PROTOCOL=TCP)(HOST=10.0.0.1)
(PORT=1446))"
```

PROTOPROC_TRACE

This parameter designates the trace facility and level of detail for tracing of the DRDA Protocol Processor. All initial Oracle Database Provider for DRDA sessions run with this setting. The parameter consists of a value pair that represents facility and level values. Multiple values may be specified simultaneously if the value tuples are separated by a comma.

It also designates the initial level of trace under which all AS session threads execute. `PROTOPROC_TRACE` is a decimal number or textual designated equivalent.

The facility names used with PROTOPROC_TRACE parameter are as follows:

- `TASK` – Task-specific operations

- NET – Network-specific operations
- SQL – SQL-specific operations
- OCI – OCI resource operations
- MEM – Memory resource operations
- ALL – All facilities mentioned already

The values of PROTOPROC_TRACE level are additive. For example, setting ERROR(4) includes WARN(2) and INFO(1) messages. The following values are expected:

- 0 or NONE – No trace is generated; this is the default.
- 1 or INFO – Minimal trace is generated.
- 2 or WARN – Warning information is generated.
- 4 or ERROR – Error information is generated.
- 8 or ADMIN – Administration information is generated.
- 255 or ALL – All details are generated.

Default Value

0 or none

Allowable Values

Facility name followed by level.

Usage Example

```
PROTOPROC_TRACE="ALL ADMIN"
PROTOPROC_TRACE="TASK WARN, NET ADMIN, MEM INFO"
```

PROTOPROC_OPTIONS

The PROTOPROC_OPTIONS parameter specifies protoproc processing options, effecting DRDA protocol operations.

Default Value

There is no default value for this parameter.

Allowable Values

The following values may be specified:

- QRYDTA/ELMODE

  Enables Extended Length Mode for all returned Query Data objects. Generally, this is used with queries that do not involve LOBs.

- EXTDTA/ELMODE

  Enables Extended Length Mode for all returned Extended Data objects. Typically used with queries involving LOBs.

- ALLDSS/ELMODE

Enables Extended Length Mode for all protocol objects. This is an alternative to either above value, and should not be specified unless other values have no positive effect.

- `LOGNAME/<NAME>`

  Specifies a static DRDA Log Name. The Log Name should alphanumeric and consist of 18 characters. This option may be required when you use DB2 for z/OS as a client.

- `LOGTSTMP/<YYYYMMDDHHMMSSTTTT>`

  Specifies a static DRDA Log Timestamp. The timestamp must have the following format:

  `YYYYMMDDHHMMSSTTTT`

  This value may be required when you use DB2 for z/OS as a client.

Syntax

```
PROTOPROC_OPTIONS = value {, value ...}
```

Usage Example

```
PROTOPROC_OPTIONS = QRYDTA/ELMODE, EXTDTA/ELMODE, LOGNAME/ORACLEDB,
LOGTSTMP/201609191201020001
```

> **See Also:**
>
> Extended Length Mode section, in Chapter 12 — Restrictions on Using Oracle Database Provider for DRDA

4

SQL Translation and Examples for Oracle Database Provider for DRDA

SQL Translation is a new feature in Oracle Database 12*c* Release 2 (12.2). Oracle Provider for DRDA may be used without SQL Translation with earlier releases of Oracle Database.

For more information on SQL Translation, see *Oracle® Database Migration Guide*.

> **Note:**
>
> SQL Translation is an optional feature and recommended in cases where SQL used within the application is more DB2–centric syntax.

Overview of SQL Translation Process

Oracle Database Release 12*c* introduces the concept of SQL Translation. This feature enables the translation of 'foreign' SQL statements, such as DB2, into a SQL syntax that may be correctly used by Oracle Database. SQL Translation itself is implemented through a SQL Translator that is most often supplied by a third party to the translation. The SQL Translator inspects the input SQL, and sometimes alters it to adhere to Oracle SQL syntax. A SQL Translation Profile, which is specified through a SQL Translator Interface Package, specifies a SQL Translator that is used.

A SQL Translation Profile is a schema-level object of type `SQL TRANSLATION PROFILE`. It references a PL/SQL package through its `ATTR_TRANSLATOR` attribute; this package is known as the SQL Translator Interface Package. The package specifies the third-party SQL translator that performs the SQL translation when the SQL Translation Profile is active. Only one SQL Translation Profile may be enabled at a time.

SQL Translation Profiles may be shared among users. Commonly, all users share the same single SQL Translation Profile for a set of packages, but that is not necessary.

Note that each DB2 package may be associated with a SQL Translation profile through the attributes kept for that package. The SQL Translator associated with the SQL Translation Profile specified for the DB2 package is used when preparing SQL statements within that DB2 package.

Implementing SQL Translation

In order for translation to proceed, the following sequence of events must take place:

1. Acquisition of a SQL Translator.
2. Creation of a SQL Interface Package that references that translator.
3. Creation of a SQL Translation Profile that references the SQL Interface Package.

This step may be done only once in the life of an instance. However, it must be performed at least once to use SQL Translation.

In situations with multiple translators, or where different SQL Translation Profiles are necessary, this process may be repeated.

4. Association of DB2 packages with a SQL Translation profile.

   This step must be completed for each package created.

   Note that a package does not have to be created before it is associated with the translation profile; only the name of the package is necessary. This step does not validate that a particular package already exists.

5. At execution time, the user passes SQL text to Oracle Database Provider for DRDA through the package.
6. When Oracle Database Provider for DRDA acquires SQL text, it checks if the package is associated with a SQL Translation Profile, and then sets that SQL Translation Profile to be in effect during the time when SQL text is parsed and executed.
7. After Oracle Database Provider for DRDA prepares SQL text for execution, Oracle Database uses the current SQL Translation Profile to translate the SQL statements, and then executes them.

Requirements for SQL Translation

Successful SQL translation may occur only in the following are true:

- A SQL Translation Profile must be enabled for the session, through the following command:

  ```
  ALTER SESSION SET SQL_TRANSLATION_PROFILE
  ```

- The process must specify that incoming SQL statements are in a foreign syntax, or in a non-Oracle SQL dialect. In all cases discussed here, these dialect are variants of DB2 SQL.

For Oracle Database Provider for DRDA product, the preceding two conditions are coupled; if a DB2 package is associated with a SQL Translation Profile, then the SQL statements are expected to be in a foreign syntax, and the SQL Translator associated with the SQL Translation Profile is called to translate any SQL in that package.

Specifics of Translating DB2-Specific SQL Syntax

While most of the SQL constructs that a client application submits to Oracle Database Provider for DRDA may be executed directly, some DB2 SQL constructs are not recognized by Oracle. Consider the known issues that occur when translating DB2 SQL statements issued by an application that is re-configured to use an Oracle Database instance.

If a SQL Translation Profile is in place, the SQL Translator associated with the profile may be designed to alter these SQL statements so that the application performs equivalent or similar operations in Oracle SQL, and returns the expected results.

DB2 Special Registers

Oracle Database does not support the `CURRENT TIME` special register construct, to get the current time of day. Calls to this construct, as in the following example, results in an `ORA-00936` error.

```
SELECT CURRENT TIME FROM SYSIBM.SYSDUMMY1
```

DB2 SQL Functions and Procedures

Oracle does not support some functions that are defined in DB2. For example, the `CEILING` function does not exist in Oracle; instead, Oracle SQL syntax includes a compatible `CEIL` function.

DB2 Named Datatypes

Some elementary SQL datatypes, such as `BIGINT`, are not defined in Oracle. When the application runs against Oracle, casting a column value or constant as a `BIGINT` produces an error. The following example results in an `ORA-00902` error because `BIGINT` is not recognized as a valid Oracle datatype.

```
SELECT CAST(12345678912 AS BIGINT) from SYSIBM.SYSDUMMY1
```

DB2 Syntactic Statements

An `INSERT` statement in DB2 may contain syntactic clauses that are Oracle does not interpret. An example of such is the *isolation* clause, shown in the following code example.

```
INSERT INTO SCOTT.DEPT VALUES(50, "FARMING", "SPRINGFIELD") WITH CHG
```

SQL Translator Interface Package

A SQL Translation Profile is a schema-level object of type `SQL TRANSLATION PROFILE` created through the `DBMS_SQL_TRANSLATOR.CREATE_PROFILE()` procedure. The SQL Translator Interface Package is a PL/SQL package of a certain format; it references the third-party-supplied translator objects and is, itself, referenced by the SQL Translation Profile. So the SQL Translator Interface Package connects the SQL Translation Profile and the third-party supplied SQL translator objects.

About SQL Translator Interface Package

A SQL Translation Profile references a PL/SQL wrapper package that has a fixed format, the ***SQL Translator Interface Package***. When a session sets a `SQL TRANSLATION PROFILE`, it specifies that all SQL is translated by the third-party SQL translator associated with the SQL Translator Interface Package. The procedure `translate_sql()` of the SQL Translator Interface Package performs the translation.

Note that Oracle **does not** provide a SQL Translator. Instead, a SQL Translator must be obtained from third-party vendors, or developed internally. Oracle provides various administrative scripts for creating and managing a SQL Translation Profile.

Creating a SQL Translator Interface Package

Example 4-1 shows a simple SQL Translation Interface Package used with a SQL Translation Profile. The language and name specifications are relative to the language-type and callable-names in the third-party SQL translator. After logging into the Oracle Database with SYSDBA privileges, the following package declaration must be made. The package name is the value of the TRANSLATOR_ATTR attribute of the SQL Translation Profile.

Example 4-1 Creating a SQL Translator Interface Package

```
create or replace package SYSIBM.DBTooSQLTranslator as
  procedure translate_sql(
    sql_text in CLOB,
   translated_text out CLOB);
  procedure translate_error(
    error_code in BINARY_INTEGER,
    translated_code out BINARY_INTEGER,
    translated_sqlstate out VARCHAR2);
end;
/
create or replace package body SYSIBM.DBTooSQLTranslator as
  procedure translate_sql(
    sql_text in CLOB,
    translated_text out CLOB )
    as language JAVA
  name /* actually the "signature" of the third-party callable */
  /* procedure associated with translate_sql    */
   'DBTooSQLApiInterface.translateSQL(oracle.sql.CLOB, oracle.sql.CLOB[])';

   procedure translate_error(error_code in BINARY_INTEGER,
    translated_code out BINARY_INTEGER,
    translated_sqlstate out VARCHAR2) as
  language JAVA
  name /* actually the "signature" of the third-party callable */
  /* procedure associated with translate_error  */
 'DBTooSQLApiInterface.translateError(oracle.sql.CLOB, oracle.sql.CLOB[])';
 end;
 /
```

Granting EXECUTE Access to SQL Translator Interface Package

Because the SQL Translator Interface Package is called at run-time, it must have EXECUTE access enabled. Example 4-2 shows how to grant this access.

Example 4-2 Granting EXECUTE access to SQL Translator Interface Package

```
GRANT EXECUTE ON SYSIBM.DBTooSQLTranslator TO DRDAAS_USER_ROLE
```

Creating a SQL Translation Profile

The SQL Translation Profile may be created and administered by any user who has the CREATE SQL TRANSLATION PROFILE authority and TRANSLATE ANY SQL authority. The section on Granting Required Authority to Users with DRDAAS_TRANS_ADMIN Role shows how to grant these two privileges to DRDAAS_TRANS_ADMIN. These privileges may be granted by a user with existing SYSDBA privileges.

The ADMIN OPTION clause enables DRDAAS_TRANS_ADMIN to GRANT the TRANSLATE ANY SQL authority to other Oracle users. In this manner, the DRDAAS_TRANS_ADMIN may allow many users with DRDAAS_USER_ROLE to use the translation facility, as demonstrated in the section Granting Translation Authority to Users with DRDAAS_USER_ROLE.

The actual SQL Translation Profile may be managed through a script provided in Creating and Managing the SQL Translation Profile. Note that the administering id must already have the required authority to perform CREATE SQL TRANSLATION PROFILE.

Related Topics

- Granting Translation Authority Through Administrator Role
- Granting Translation Authority Through User Role
- Creating and Managing SQL Translation Profile

Granting Translation Authority Through Administrator Role

Example 4-3 Granting Required Authority to Users with DRDAAS_TRANS_ADMIN Role

```
GRANT CREATE SQL TRANSLATION PROFILE TO DRDAAS_TRANS_ADMIN;
GRANT TRANSLATE ANY SQL TO DRDAAS_TRANS_ADMIN WITH ADMIN OPTION;
```

Granting Translation Authority Through User Role

Example 4-4 Granting Translation Authority to Users with DRDAAS_USER_ROLE

```
GRANT TRANSLATE ANY SQL TO DRDAAS_USER_ROLE;
```

Creating and Managing SQL Translation Profile

Example 4-5 Creating and Managing the SQL Translation Profile

```
declare
  PROFILE_DOES_NOT_EXIST exception;
  pragma EXCEPTION_INIT(PROFILE_DOES_NOT_EXIST, -24252);
  /* profile_name is the nsme of the SQL Translation Profile */
  /* created here.  */
  profile_name VARCHAR2(32) := 'DRDAAS_TRANS_ADMIN.MY_PROFILE';

  /* SYSIBM is the schema in which the SQL Translator Interface */
  /* package (viz., SYSIBM.DBTooSQLTranslator) is found.   */
  sql_trnsltr_intfc_schema VARCHAR2(32) := 'SYSIBM';

  /* DBTooTranslator is the unqualified package name of the SQL */
  /* Translator Interface Package */
  sql_trnsltr_intfc_pkgnm VARCHAR2(32) := 'DBTooSQLTranslator';

  sql_trnsltr_intfc_pkg VARCHAR2(128);
  grant_cmd VARCHAR2(256);
  cursor_id NUMBER;

begin
  sql_trnsltr_intfc_pkg := sql_trnsltr_intfc_schema || '.'  ||
    sql_trnsltr_intfc_pkgnm;
```

```
  begin
    DBMS_SQL_TRANSLATOR.DROP_PROFILE(profile_name);
    exception
      WHEN PROFILE_DOES_NOT_EXIST THEN NULL; /* ignore if non-existant */
  end;
  /* Create SQL Translation Profile */
  DBMS_SQL_TRANSLATOR.CREATE_PROFILE(profile_name);
  /* Associate the SQL Translator Interface Package denoted by */
  /* sql_trnsltr_intfc_pkg with this profile    */
  DBMS_SQL_TRANSLATOR.SET_ATTRIBUTE(profile_name,
    DBMS_SQL_TRANSLATOR.ATTR_TRANSLATOR,
    sql_trnsltr_intfc_pkg);
  /* Mark this SQL Translation Profile as "registered" */
  DBMS_SQL_TRANSLATOR.SET_ATTRIBUTE(profile_name,
    DBMS_SQL_TRANSLATOR.ATTR_TRANSLATION_REGISTRATION,
    DBMS_SQL_TRANSLATOR.ATTR_VALUE_TRUE);
  /* The owner of the SQL Translator Interface Package must have */
  /* full authority for the SQL TRANSLATION PROFILE    */
  grant_cmd := 'GRANT ALL ON SQL TRANSLATION PROFILE ' ||
    profile_name || ' TO ' || sql_trnsltr_intfc_schema;
  cursor_id := DBMS_SQL.OPEN_CURSOR();
  DBMS_SQL.PARSE(cursor_id, grant_cmd, DBMS_SQL.NATIVE);
  DBMS_SQL.CLOSE_CURSOR(cursor_id);
  /* Let all with DRDAAS_USER_ROLE have access to the SQL Translation profile.  */
  grant_cmd := 'GRANT USE ON SQL TRANSLATION PROFILE ' ||
    profile_name || ' TO DRDAAS_USER_ROLE';
  cursor_id := DBMS_SQL.OPEN_CURSOR();
  DBMS_SQL.PARSE(cursor_id, grant_cmd, DBMS_SQL.NATIVE);
  DBMS_SQL.CLOSE_CURSOR(cursor_id);
end;
/
```

Using Third-Party SQL Translators

To use a third-party translator, its files and objects must be installed in the directory $*ORACLE_HOME*/rdbms/drdaas/jlib.

In case of difficulties, use `DBMS_JAVA.SET_OUTPUT()` procedure to redirect server-side error messages to `DBMS_OUTPUT`.

For more information on using Java in Oracle, refer *Oracle® Database Java Developer's Guide*.

Using a Third-Party SQL Translator, Loaded as a Single Object

If the third-party SQL translator is in Java, Example 4-6 may be run in SQL*Plus environment by a `SYSDBA` user. Example 4-6 uses `DBMS_JAVA.LOADJAVA()` procedure to load the objects into the `SYSIBM` schema; it loads a single third-party object, `DBTooSQLAPI.jar`.

Example 4-6 Loading a Third-Party SQL Translator; Single Object

```
begin
  DBMS_JAVA.LOADJAVA('-definer -genmissing -schema SYSIBM ' || '
  ' rdbms/drdaas/jlib/DBTooSQLAPI.jar',
  '((* SYSIBM)(* PUBLIC)(* -))');
end;
/
```

Using a Third-Party SQL Translator, Loaded as Multiple Objects

If the third-party translator consists of multiple objects, each component must be specified in the `LOADJAVA` call. Example 4-7 specifies two translator objects, `DBTooSQLAPI.jar` and `DBTooMainClass.class`.

Example 4-7 Loading a Third-Party SQL Translator; Multiple Objects

```
begin
  DBMS_JAVA.LOADJAVA('-definer -genmissing -schema SYSIBM ' ||
  ' rdbms/drdaas/jlib/DBTooMainClass.class' ||
  ' rdbms/drdaas/jlib/DBTooSQLAPI.jar',
  '((* SYSIBM)(* PUBLIC)(* -))');
end;
/
```

Using a Translator Management Script

Oracle Database ships a `drdasqtt_translator_setup.sql` script, which manages translation profiles. The script must be invoked in SQL*Plus by a user with `SYSDBA` privileges. It asks for the following inputs:

1. SQL Translator Interface Package Schema, such as `SYSIBM`.
2. SQL Translator Interface Package Name, such as `DBTooTranslator`.
3. SQL Translation Profile Schema, such as `DRDAAS_TRANS_ADMIN`.
4. SQL Translation Profile Name, such as `MY_PROFILE`.
5. Language type of the third-party translator, such as Java.
6. The names of files or objects supplied by the third-party vendor. If more than one is supplied, enclose the list in ***four (4)*** single quotes and separate the items by blank spaces, as in the following code:

   ```
   ''''rdbms/drdaas/jlib/DBTooMainClass.class rdbms/drdaas/jlib/DBTooSQLAPI.jar''''
   ```

7. The signature (entry name plus argument descriptions) of the entry in the files or objects supplied by the third-party vendor that are used for translating SQL. For Java-based third-party code, the signature may be obtained through the `javap` program. Note that signatures that contain blank space must be enclosed within double quotes.
8. The signature (entry name plus argument descriptions) for the entry in the files or objects supplied by the third-party vendor that are used for translating error codes. For Java-based third-party code, the signature may be obtained through the `javap` program. Note that signatures that contain blank space must be enclosed within double quotes.

Verifying the SQL Translator Profile

The following steps verify that the SQL Translation Profile is correctly installed and fully enabled.

To verify the SQL Translator Profile configuration:

1. Log into Oracle Database with SYSDBA privileges

2. Check that the translator profile is loaded into Oracle Database.

   ```
   SELECT * FROM ALL_SQL_TRANSLATION_PROFILES;
   ```

3. Log in with an id that has `DRDAAS_USER_ROLE` privileges.
4. Ensure that the role is set:

   ```
   SET ROLE DRDAAS_USER_ROLE;
   ```

5. Set the SQL Translation Profile for of session to the value specified at the time the SQL Translation Profile was created.

   ```
   ALTER SESSION SET SQL_TRANSLATION_PROFILE = DRDAAS_TRANS_ADMIN.MY_PROFILE;
   ```

6. Attempt the following commands:

   ```
   ALTER SESSION SET EVENTS = '10601 trace name context forever, level 32';
   SELECT CAST(1234567 AS BIGINT) FROM DUAL;
   ALTER SESSION SET EVENTS = '10601 trace name context off';
   SELECT CAST(1234567 AS BIGINT) FROM DUAL;
   ```

   The first `SELECT` should succeed, while the second should fail. The `ALTER SESSION SET EVENTS` commands specify that the following SQL is one of:

   - foreign syntax (trace name context forever, level 32)
   - native Oracle syntax (trace name context off)

   This works only when using SQL*Plus.

Altering the SQL Translation Profile

At times, it becomes necessary to completely change the SQL Translation Profile, and make the `SQL TRANSLATION PROFILE` attribute of a DB2 package reference a new SQL Translation Profile.

DB2 packages usually come in sets, and the names of the DB2 packages are determined by the client. However, if the client uses ODBC to access Oracle Database Provider for DRDA, the ODBC driver determines the names of the packages.

Oracle supplies two scripts that may be used to set the SQL Translation Profile attribute for a set of packages.

- If the DataDirect ODBC driver accesses Oracle Database Provider for DRDA, use the `drdasqt_set_profile_dd.sql` script, in the `drdaas/admin` directory.
- If the IBM ODBC driver accesses Oracle Database Provider for DRDA, use the `drdasqt_set_profile_ibm.sql` script, in the `drdaas/admin` directory. Native client application may also use this script, but it may have to be extended.

These scripts may be copied and altered for use with other sets of DB2 packages.

Additionally, each of these scripts must be run in SQL*Plus by a user with `DRDAAS_ADMIN_ROLE` privileges. The script prompts for the qualified name of the profile that is referenced by the various packages (such as `DRDAAS_TRANS_ADMIN.MY_PROFILE`). It also prompts for the default Package Collection schema, which is usually `NULLID`.

5

Administration and Customization of Oracle Database Provider for DRDA

Consider various administration and customization issues.

Migration Steps using Oracle Database Provider for DRDA

While migration of existing DB2 applications to Oracle Database is data- and target-specific, the general methodology has the following six steps:

1. Installing and configuring Oracle Database Provider for DRDA software
2. Installing Oracle Database Provider for DRDA objects in the Oracle Database
3. Administering DRDA Package authority
4. Migrating DB2 data
5. Re-targeting the application
6. Tuning SQL Translation and Datatypes

Considerations for Using Oracle Database Provider for DRDA

Before installing Oracle Database Provider for DRDA software, an organization must consider several operational and resource issues. Flexibility and performance of machine and network resources is paramount when determining whether an optimal installation is as a standalone Oracle home, an Oracle home within an existing Oracle Database, or on a machine that is entirely separate from the Oracle Database. Additionally, the nature of all possible DB2 clients that must use the installation is a determining factor; in this context, DB2 is considered a client.

Related Topics

- installing Oracle Database Provider for DRDA

Prerequisites to Installing Oracle Database Provider for DRDA

Before installing Oracle Database Provider for DRDA objects in the Oracle Database, one or more users must be designated as DRDA Administrators, and have the Administrator role.

Similarly, designate users who will be accessing the Oracle Database through Oracle Database Provider for DRDA or DB2 applications, and grant to them roles and privileges of DRDA User. Some aspects of setting the DRDA User's authority and configuration may need to be delayed until further in the migration process. This

mostly concerns specific DRDA packages used by the application, and any specific SQL translations or datatype tuning. If the application's packages are identified before migration, these may be applied as part of the package authorization workflow.

Related Topics

- Administrator Role
- User Role

Administering DRDA Package Authority

In order to successfully access Oracle Database from DRDA or DB2 applications through Oracle Database Provider for DRDA, package authorization must be in place. At a minimum, the following information must be collected about the application and its users:

- package collection ID, such as `NULLID`
- package name, such as `DSNPBD3`
- package version name, if applicable, such as `01` or `NULL`
- name of the Oracle user who must access the database, such as `DRDAUSR`

A SQL Translation Profile Name must also be designated for the application represented by the package.

Related Topics

- SQL Translator Interface Package
- Packages

Migrating DB2 Data

In DB2, objects may be created under an arbitrary schema, whereas schema names are not arbitrary in Oracle Database. Therefore, careful use of schemas must be considered when migrating data from DB2 to Oracle. In Oracle, all schema objects, such as tables, views, synonyms, and so on, must be allocated in a schema of an actual user. This obviously effects how these objects are named, created, and accessed.

Consider the following example: `USER1` creates tables `"USER1"."TABLE1"` and `"USER2"."TABLE2"`. In DB2, `TABLE1` and `TABLE2` are owned by `USER1`, because `USER1` is their creator. In Oracle, the table `"USER2"."TABLE2"` is owned by user `USER2`. Additionally, `USER1` could not have created `TABLE2` unless `USER1` has `CREATE ANY TABLE` privelege. Instead, `USER2` must create `TABLE2`, and then grant `USER1` access to it.

Data migrated from DB2 to Oracle must be defined also in terms of Oracle datatypes. While Oracle uses ANSI-defined datatype names, they do not necessarily have the same range limits or semantics as the DB2 implementation. To accurately model existing DB2 application datatypes, review the section Data Dictionary for Oracle Database Provider for DRDA.

After creating the schema and objects with appropriate datatypes, the data may be imported into Oracle.

Related Topics

- Data Dictionary Emulation in Oracle Database Provider for DRDA

Retargeting the Application to Use Oracle Database

The following example shows how to migrate DB2 z/OS applications. You would need to follow similar steps when migrating DB2/LUW or DB2/400 applications. Refer to IBM documentation for details of each product's equivalent steps.

> **Note:**
>
> This example is applicable only for DB2 for z/OS.

There are two general categories of applications: ***native applications*** and ***remote applications***.

Re-targeting Native Applications

Typical DB2 applications are called native because they interact with a local DB2 system directly, through an internal IPC mechanism. These applications use embedded SQL programming, and utilize the DB2 SQL PreProcessor. Pre-processing the source generates an ***execution plan*** that is stored in a Database Resource Module (DBRM). Users must upload, or bind the execution plan to the local DB2 instance before the program runs.

The execution plan contains all the static SQL embedded in the application source, as well as additional attributes such as ***location***, also called the Current Server. By default, Current Server is blank; this indicates that the server is on the local DB2 instance. It is possible, however, to re-target the execution plan to run all operations on another server by setting a new value for the Current Server attribute.

The following steps should be performed by an IBM DB administrator.

1. Create location entries in the DB2 Communications Database.

   DB2 has a internal communications system for connecting to remote DB2 instances. To address a remote instance, insert records into the `SYSIBM.IPNAMES` table, the `SYSIBM.LOCATIONS` table and, optionally, into the `SYSIBM.USERNAMES` table.

   See IBM DB2 documentation for a description of the DB2 Communications Database facility.

   The following command inserts a linkname `REMHOST`, a location entry `DRDAAS`, and an optional username mapping entry in the DB2 Communications Database. The linkname specifies the hostname or IP address of the computer that is running Oracle Database Provider for DRDA. The location specifies an RDB name that uses the linkname and the port number that Oracle Database Provider for DRDA is listening on. These correspond to Oracle Database Provider for DRDA configuration parameters `DATA_PORT` and `RDB_MAP`. Note that the location name must match exactly to the `RDB()` value specified in the `RDB_MAP` parameter.

   ```
   INSERT INTO SYSIBM.IPNAMES (LINKNAME,SECURITY_OUT,USERNAMES,IPADDR)
     VALUES ('REMHOST','P','O','remotehost.remotedomain.com');
   ```

```
INSERT INTO SYSIBM.LOCATIONS (LOCATION,LINKNAME,PORT)
  VALUES ('DRDAAS','REHMOST','1446');

INSERT INTO SYSIBM.USERNAMES (TYPE,AUTHID,LINKNAME,NEWAUTHID,PASSWORD)
  VALUES ('O',' ','REMHOST','DRDAUSER', 'userpwd' );
```

2. Remotely bind the application Plan to Oracle Database Provider for DRDA.

   After the location entries are inserted, you must remotely bind the application execution plan. The following code binds plan `DSNPBD3` through the DSN command processor `IKJEFT01`. Note that location `DRDAAS` prefixes the collection Id.

   ```
   BIND PACKAGE(DRDAAS.NULLID) MEMBER(DSNPBD3) -
              ACT(REP) ISO(CS) CURRENTDATA(YES) ENCODING(EBCDIC)
   ```

3. Locally bind the package with current server.

   After the plan is bound remotely, re-bind the local plan using the current server option to re-target execution. The following code binds plan `DSNPBD3` through the DSN command processor `IKJEFT01`.

   Note that the plan must be referred to in the remote plan through the package list, `PKLIST`, and must specify both the location in the package reference, `DRDAAS.NULLID.DSNPBD3`, and specify the `CURRENTSERVER` option that contains the location.

   ```
   BIND PLAN(DSNPBD3) -
      PKLIST(DRDAAS.NULLID.DSNPBD3) -
      ACT(REP) ISO(CS) CURRENTDATA(YES) ENCODING(EBCDIC) -
      CURRENTSERVER(DRDAAS)
   ```

4. After the plan is bound remotely and re-bound locally, the application runs using plan `DSNPBD3`, implicitly makes a remote connection through the local DB2 to Oracle Database Provider for DRDA, and executes all operations of the plan remotely. The local DB2 remains a pass through coordinator in this configuration.

Re-targeting Remote Applications

Remote applications are typically not directly tied to the Local DB2. Such applications typically are referred to as being *network-aware* or *network-oriented* and have a remote server location configuration attribute that is used to specify what and where to connect to.

Such applications utilize Oracle Database Provider for DRDA through the network protocol. Re-targeting of this type of application is simple to configure, as the following steps show.

1. Change the configuration options of the Application to use the hostname (or IP address), port number and RDB name of that configured in Oracle Database Provider for DRDA. An example of this is through ODBC, in which the DSN entry contains network parameters.

   In this example, the `Network` and `PortNumber` parameters correspond to the `Linkname` and `Location` entries inserted into the DB2 Communication Database example used earlier. The `Database` parameter corresponds to the `Location` name. All of which, again, correspond to the `DATA_PORT` and `RDB_MAP` parameters of the configured Oracle Database Provider for DRDA.

   Here is an example of an `odbc.ini` file.

```
[DRDAAS]
       Network=remotehost.remotedomain.com
       PortNumber=1446
       Database=DRDAAS
```

2. Execute the package resource binding operation for the application.

   Often this is handled implicitly by the application itself, or is documented as a one-time step to setting up the applications access and resource to a remote DB2 instance. Refer to the documentation for the specific application for Binding instructions.

Translating SQL Statement and Typing Datatypes

Some applications may have DB2-specific SQL that is beyond the automatic translation mechanism of SQL translation, or may be expecting a very specific datatype for a particular column in a query. In such cases it may be necessary to manually insert SQL substitution statements, or add item-specific datatype manipulations.

For example, suppose an application has a specific SQL statement that has the following DB2-specific syntax: `SELECT LOG2(COL1) FROM TABLE1`. To work correctly in Oracle, the SQL needs to be translated into this statement: `SELECT LOG10(COL1,2) FROM TABLE1`.

Through SQL Translation's Register facility, a direct translation may be registered for this SQL statement, as shown in the section Registering a SQL Substitution Statement. Note that this must be done by the user who is executing the SQL statement; remember that the SQL Translation Profile must created as a resource for that user.

After the SQL translator is registered, when the application issues the original SQL it is implicitly translated to the new SQL and processes.

In some very specific cases, application clients require the datatypes of select items in a query to be returned in a very specific format.

Let's say that the result of the translated query `SELECT LOG10(COL1,2) FROM TABLE1` returns a `DECFLOAT34` datatype, but the application is unable to process it, it is possible to implicitly coerce the datatype to another, compatible type.

If the application supports the `DOUBLE PRECISION` datatype, it is possible to use the `TYPEMAP` facility to add this specific coercion described in Registering an On-Demand Datatype Conversion.

Registering a SQL Substitution Statement

Example 5-1 Registering a SQL Substitution Statement

The application's package has been assigned the Profile name `DB2ZOS`.

```
connect DRDAUSER/userpwd
execute dbms_sql_translator.register_sql_translation('DB2ZOS',
   'SELECT LOG2(COL1) FROM TABLE1',
   'SELECT LOG10(COL1,2) FROM TABLE1')
```

Registering an On-Demand Datatype Conversion

Example 5-2 Registering an On-demand Datatype Conversion

```
connect DRDAADM/adminpwd

execute DBMS_DRDAAS_ADMIN.SET_TYPEMAP('NULLID', 'DSNPBD3', NULL,
   'TABLE1:LOG10(COL1,2)', 'NUMBER=DOUBLE')

execute DBMS_DRDAAS_ADMIN.SET_TYPEMAP('NULLID', 'DSNPBD3', NULL,
   'TABLE1:LOG10(COL1,2)', 'NUMBER(0,-127)=DOUBLE')
```

6

Diagnostics and Maintenance of Oracle Database Provider for DRDA

Consider the issues of Diagnostics and Maintenance.

Diagnostics for Oracle Database Provider for DRDA

Diagnostics for Oracle Database Provider for DRDA consist of the trace facility, which is configurable through the trace level before the application runs, and may be adjusted by the command-line tool. Oracle Database Provider for DRDA uses the Automatic Diagnostic Repository (ADR) to hold all logs, traces and dump records. See section "Automatic Diagnostic Repository (ADR)" in *Oracle® Database Administrator's Guide*.

The trace logs DRDA protocol errors before, during, and after each client session. There are specific DRDA architecture error alerts that are intended to diagnose a protocol violation. Additional errors from the OCI session may also be logged there.

Diagnostics may be summarized as the ability to trace, or collect diagnostic information, usually in a file.

The trace directory may be specified by the user. By default, Oracle Database Provider for DRDA creates a trace directory in *`ORACLE_BASE/diag`*or `ORACLE_HOME/log/diag`. Within this root directory, Oracle Database Provider for DRDA stores trace files in `dps/`*`drdaas`*`/instance/trace`, where *`instance`* is the instance name used in the `drdaas.ora` file. The `adrci` utility enables viewing and manipulation of trace files.

Depending on the application, different levels of diagnostic detail might become necessary. By default, diagnostic depth is set to off, for performance reasons. However, incidents are logged in the repository if the AS fails.

Related Topics

- Command-line Utility

Maintaining Oracle Database Provider for DRDA

To communicate with the Application Server, Oracle provides an external command interface. The command interface supports the following interactions:

- Starting the server
- Stopping the server
- Determining server status
- Displaying connected client sessions
- Displaying details of client sessions, which includes:

  - Session state (command being executed, such as preparing, executing, fetching, idle, and so on)
  - Last SQL statement prepared
  - Client IP address and port number
  - Oracle SQL Session Id
- Pausing a SQL session
- Terminating a SQL session
- Reloading server configuration

See “Command-line Utility for Oracle Database Provider for DRDA” for details of these operations.

Related Topics

- Command-line Utility for Oracle Database Provider for DRDA

7

Datatype Support and Conversion in Oracle Database Provider for DRDA

Consider datatype support in Oracle, and conversion between Oracle and DRDA datatypes.

Overview of Datatype Conversion

DRDA utilizes *Formatted Data Object Content Architecture (FD:OCA)* for datatype encoding. Several types do not have a direct analog to Oracle native types, and require conversion. Also, some Oracle datatypes have no direct encoding support in FD:OCA.For example, consider Oracle `NUMBER`, which may contain a wide range of values, both *integers* and *floating point*. This duality prevents it from being mapped to a specific DRDA type, to mitigate loss of value of the number. Any choice of type will have some loss of either precision or scale at extreme ranges of value.

There are two datatype conversions used by Oracle Database Provider for DRDA: conversion of DRDA MetaData Descriptors to Oracle OCI interface types, and conversion of Oracle column types to DRDA MetaData Descriptors. For application programmers, these are described through the SQL Type of the bind variable or described column type. See sections “Conversion between DRDA Datatypes to Oracle Datatypes” and “Conversion of Oracle Datatype to DRDA.”

A general mechanism for mapping Oracle `NUMBER` is covered in section “Datatype Equivalence and Remapping”.

Related Topics

- Conversion between DRDA Datatypes to Oracle Datatypes
- Conversion of Oracle Datatype to DRDA
- Datatype Equivalence and Remapping

Numerical Range Considerations; General

When converting between Oracle `NUMBER`, IEEE floating point, IBM Hexadecimal floating point (HEX floating point, S390 or System390 floating point), and Decimal floating point (`DECFLOAT`) datatypes, note that they have different ranges and capabilities. For example, all values of `IBM HEX FLOAT` bind variables in a client-side program fit in an Oracle `NUMBER`, but not all values of Oracle `NUMBER` may be returned correctly in an `IBM HEX FLOAT`; `DECFLOAT34` is a better choice.

Some other considerations include the following:

- **Infinities.** Some floating point types support positive and negative infinities.

  When infinities are used for datatypes that don't support them, the highest possible number for positive infinities and its negative for negative infinities is used.

- **Floating Point.** IEEE `FLOAT` columns may be defined in Oracle with types of `BINARY_FLOAT` and `BINARY_DOUBLE`. In DB2 z/OS the floating point types (`REAL`, `FLOAT`, `DOUBLE` and `DOUBLE PRECISION`) are IBM HEX floating point. In DB2/400 and DB2 LUW, the floating point types (`REAL`, `FLOAT`, `DOUBLE` and `DOUBLE PRECISION`) are IEEE floating point.
- **Not a Number.** Some datatypes support Not A Number (NAN), a special value to indicate either that no value was assigned, or the result of a computation is invalid or undefined.

Oracle NUMBER

Oracle `NUMBER` has the following characteristics:

Lower Range

1E-130

Upper Range

9.999 999 999 999 999 999 999 999 999 999 999 999 9E+125

Infinity

Supported for both negative and positive infinity

Not A Number

Not supported

FLOAT (IBM HEX or S390)

The following characteristics apply to `FLOAT`, `DOUBLE` and `LONG DOUBLE` sub datatypes.

Lower Range

5.397605×10^{-79}

Upper Range

$7.237005 \times 10^{+75}$

Infinity

Not supported

Not A Number

Not supported

FLOAT (IEEE)

The following characteristics apply to `FLOAT` (Oracle `BINARY_FLOAT`), `DOUBLE` (Oracle `BINARY_DOUBLE`), and `LONG DOUBLE` sub datatypes.

Infinity

Supported for both positive and negative infinity

Not A Number

Supported

The bounds for the subtypes follow:

Lower Range

`FLOAT` (Oracle `BINARY_FLOAT`): 1.175 494 x 10^{-38}

`DOUBLE` (Oracle `BINARY_DOUBLE`): 2.225 074 x 10^{-308}

`LONG DOUBLE`: 3.362 103 x 10^{-4932}

Upper Range

`FLOAT` (Oracle `BINARY_FLOAT`): 3.402 823 x 10^{+38}

`DOUBLE` (Oracle `BINARY_DOUBLE`): 1.797 693 x 10^{+308}

`LONG DOUBLE`: 1.189 731 x 10^{+4932}

DECFLOAT

The following characteristics apply to `DECFLOAT7`, `DECFLOAT16`, and `DECFLOAT34` sub datatypes.

Infinity

Supported for both positive and negative infinity

Not A Number

Supported

The bounds for the subtypes follow:

Lower Range

`DECFLOAT7`: 0.000 001 x 10^{-95}

`DECFLOAT16`: 0.000 000 000 000 001 x 10^{-383}

`DECFLOAT34`: 0.000 000 000 000 000 000 000 000 000 001 x 10^{-6143}

Upper Range

`DECFLOAT7`: 9.999 999 x 10^{+96}

`DECFLOAT16`: 9.999 999 999 999 999 x 10^{+384}

`DECFLOAT34`: 9.999 999 999 999 999 999 999 999 999 999 x 10^{+6144}

Numerical Range Considerations, for COBOL Users

DRDA databases offer three options for integer types: `SMALLINT` (2 binary bytes), `INTEGER` (4 binary bytes), and `BIGINT` (8 binary bytes). During conversion, Oracle columns that hold equivalent values must be defined based on usage rather than on the type used in the DB2 `CREATE TABLE` definition.

The actual range of DRDA `SMALLINT`, `INTEGER` and `BIGINT` follows:

- `SMALLINT` has a lower bound of `-32,768` and an upper bound of `32,767`
- `INTEGER` has a lower bound of `-2,147,483,648` and an upper bound of `2,147,483,647`
- `BIGINT` has a lower bound of `-9,223,372,036,854,775,808` and an upper bound of `9,223,372,036,854,775,807`

However, at the level of the application, the COBOL variables that hold these DRDA column values may be declared either with a fixed number of decimal digits, or with the full binary precision of the corresponding DRDA integer datatypes.

In COBOL, the equivalent binary integer datatypes are defined as follows:

- `USAGE` of `BINARY`, `COMPUTATIONAL`, `COMP`, `COMPUTATIONAL-4`, and `COMP-4`; these are equivalent
- `PICTURE` of `S9(1-4)` for a 2-byte integer, `S9(5-9)` for a 4-byte integer, and `S9(10-18)` for an 8-byte integer.

The value is normally limited to the number of digits in the picture.

For example `PICTURE S9(4) COMP` is a 2-byte integer that normally ranges from `-32,768` to `+32,767`. However, the generated COBOL code only allows the value to range from `-9,999` to `+9,999`. When using these types of bind variables exclusively to access and update DRDA `SMALLINT`, `INTEGER`, and `BIGINT` columns, define the columns in Oracle as `NUMBER(`*n*`)`, where *n* matches the above `PICTURE S9(`*n*`)` definition.

When using `BINARY`, `COMPUTATIONAL`, `COMP`, `COMPUTATIONAL-4`, and `COMP-4` COBOL variables with the `TRUNC(BIN)` COBOL compiler option, the binary integers may range to the full bounds of the datatype. Using `COMPUTATIONAL-5` or `COMP-5` has the same effect, regardless whether the `TRUNC` compiler option is in effect. When programming in COBOL, C, PL/I, or Assembler with a full range of the binary integers, define the Oracle column as `NUMBER(`*n*`+1)`, where *n* matches the above `PICTURE S9(`*n*`)` definition.

Based on datatype and usage in DRDA, here are the recommended substitute Oracle datatypes:

Used with COBOL COMP:

- `SMALLINT` should be converted to Oracle `NUMBER(4)`
- `INTEGER` should be converted to Oracle `NUMBER(9)`
- `BIGINT` should be converted to Oracle `NUMBER(18)`

Used with COBOL COMP, TRUNC(BIN), COMP-5, C, PL/I, or Assembler binary integer variables:

- `SMALLINT` should be converted to Oracle `NUMBER(5)`
- `INTEGER` should be converted to Oracle `NUMBER(10)`
- `BIGINT` should be converted to Oracle `NUMBER(19)`

Constraining Oracle NUMBER

When using the full range of binary integer values, it is advisable to implement Oracle constraints and limit the value to the range of the corresponding datatype.

For example, a DRDA `SMALLINT` gets an equivalent Oracle `NUMBER` column that supports a full range of `SMALLINT` values, only, as demonstrated in Example 7-1.

Note however that there is a performance penalty for specifying this type of check constraint, Oracle verifies all constraints every time the column is updated.

Example 7-1 Constraining Oracle NUMBER to Exactly Match DRDA SMALLINT

```
CREATE TABLE smint_tab
  (smint NUMBER(5)
         CONSTRAINT check_smallint CHECK (smint BETWEEN -32768 AND 32767)
  )
```

Conversion between DRDA Datatypes to Oracle Datatypes

Consider the mappings between DRDA and Oracle datatypes.

Note the following abbreviations:

- In a Single Byte Character Set (SBCS), the column can only contain single byte data.
- In a Multi-Byte Character Set (MBCS), the column may contain a combination of single-byte and multi-byte characters.

INTEGER

4-byte binary number

SQL Type

496, 497

Size

4 bytes

Oracle Type

`NUMBER(10)`

SMALLINT

2-byte binary number

SQL Type

500, 501

Size

2 bytes

Oracle Type

`NUMBER(5)`

BIGINT

8-byte binary number

SQL Type

492, 493

Size

8 bytes

Oracle Type

`NUMBER(19)`

float

long double-precision (16 bytes)

SQL Type

480, 481

Range

`54 b126`

Oracle Type

`NUMBER`

DOUBLE PRECISION or FLOAT(b)

double-precision (8 bytes)

SQL Type

480, 481

Range

`22 b 53`

Oracle Type

`BINARY_DOUBLE`

REAL or FLOAT(b)

Single-precision (4 bytes)

SQL Type

480, 481

Range

`1b21`

Oracle Type

`BINARY_FLOAT`

DECIMAL(p,s)

precision and scale packed decimal

SQL Type

484, 485

Range

`1p31, 1s31`

Oracle Type

`NUMBER(p,s)`

DECIMAL(p,s) zoned

precision and scale zoned decimal

SQL Type

488, 489

Range

`1p31, 1s31`

Oracle Type

`NUMBER(p,s)`

NUMERIC(p,s)

precision and scale character decimal

SQL Type

504, 505

Range

`1p31, 1s31`

Oracle Type

`NUMBER(p,s)`

DECFLOAT(n=34)

precision and scale, with exponent; subject to loss

SQL Type

996, 997

Range

n=34

Oracle Type

`NUMBER`

DECFLOAT(n=16)

precision and scale, with exponent; subject to loss

SQL Type

996, 997

Range

n=16

Oracle Type

`NUMBER`

CHAR(n)

sbcs and mixed

SQL Type

452,453

Range

1≤ n≤255

Oracle Type

CHAR

CHAR(n) for Bit Data

byte

SQL Type

452,453

Range

1≤ n≤255

Oracle Type

RAW

VARCHAR(n)

sbcs

SQL Type

448,449

Oracle Type

VARCHAR2

VARCHAR(n)

mixed

SQL Type

448,449

Oracle Type

VARCHAR2

VARCHAR(n) for Bit Data

byte

SQL Type

448,449

Range

1≤ n≤2000

Oracle Type

`RAW`

VARCHAR(n)

sbcs

SQL Type

456,457

Range

1≤ n≤32767

Oracle Type

`LONG`

VARCHAR(n)

mixed

SQL Type

456,457

Range

1≤ n≤32767

Oracle Type

`LONG`

VARCHAR(n) for Bit Data

byte

SQL Type

456,457

Range

1≤ n≤32767

Oracle Type

`LONG RAW`

char(n+1)

sbcs

SQL Type

460,461

Range

1≤ n≤4000

Oracle Type

`CHAR`

char(n+1)

mixed

SQL Type

460,461

Range

1≤ n≤2000

Oracle Type

`CHAR`

char(n) for Bit Data

byte

SQL Type

460,461

Range

1≤ n≤2000

Oracle Type

`RAW`

VARGRAPHIC(n)

dbcs

SQL Type

464,465

Range

1≤ n≤2000

Oracle Type

NVARCHAR2

GRAPHIC(n)

dbcs

SQL Type

468,469

Range

1≤ n≤127

Oracle Type

NCHAR

VARGRAPHIC(n)

dbcs

SQL Type

472,473

Range

1≤ n≤2000

Oracle Type

NVARCHAR2

char(n) (Pascal L String)

byte

SQL Type

476,477

Range

1≤ n≤255

Oracle Type

CHAR

char(n) for Bit Data (Pascal L String)

byte

SQL Type

476,477

Range

1≤ n≤255

Oracle Type

`RAW`

DATE

Date with zero time component

SQL Type

384, 385

Oracle Type

`DATE`

TIME

Uses time component of date only, or is formatted as textual time representation

SQL Type

388, 389

Oracle Type

`DATE` or `CHAR(8)`

TIMESTAMP

Timestamp

SQL Type

392, 393

Oracle Type

`TIMESTAMP(6)`

(datalink)

no equivalent representation

SQL Type

396, 397

Oracle Type

`VARCHAR2` for sbcs, or `NVARCHAR2` for dbcs

BLOB

Binary Long OBject

SQL Type

404, 405

Oracle Type

`BLOB`

CLOB

Character Long OBject (LOB) for sbcs or mixed representation

SQL Type

408, 409

Oracle Type

`CLOB` for sbcs, and `CLOB` for mixed representation

DBCLOB

For dbcs

SQL Type

412, 413

Oracle Type

`NCLOB`

BLOB LOCATOR

Binary Long OBject (LOB)

SQL Type

960, 961

Oracle Type

`BLOB`

CLOB LOCATOR

For sbcs or mixed representation

SQL Type

964, 965

Oracle Type

`CLOB`

DBCLOB LOCATOR

For dbcs representation

SQL Type

968, 969

Oracle Type

`NCLOB`

boolean

No equivalent representation

SQL Type

2436, 2437

Oracle Type

`NUMBER(5)`

BINARY(n)

Fixed-length binary string

SQL Type

912, 913

Range

$1 \leq n \leq 255$

Oracle Type

`RAW`

VARBINARY(n)

Variable-length binary string

SQL Type

908, 909

Range

1 ≤ n ≤ 32767

Oracle Type

`LONG RAW`

XML

External form

SQL Type

988, 989

Oracle Type

`SYS.XMLType`

Conversion of Oracle Datatype to DRDA

Tables and procedures use Oracle datatypes. When describing objects, or returning data from a table or procedure, Oracle maps Oracle datatypes onto equivalent DRDA datatypes.

BINARY_FLOAT

8 bytes

SQL Type

480, 481

SQL Type Name

`DOUBLE` (8 byte floating point)

BINARY_DOUBLE

8 bytes

SQL Type

480, 481

SQL Type Name

`DOUBLE` (8 byte floating point)

VARCHAR2(n)

mixed variable length character string

SQL Type

448, 449

Range

1 ≤ n ≤ 32,767

SQL Type Name

`VARCHAR(n) FOR MIXED DATA`

LONG

Mixed long variable-length character string; Oracle `LONG` supports up to 2^31-1 bytes, but only the first `32,767` bytes are currently returned.

SQL Type

448, 449

SQL Type Name

`VARCHAR(32767) FOR MIXED DATA`

LONG RAW

Binary long variable length character string; Oracle `LONG RAW` supports up to 2^31-1 bytes, but only the first `32,767` bytes are currently returned.

SQL Type

448, 449

SQL Type Name

`VARCHAR(32767) FOR BIT DATA`

NVARCHAR2(n)

National variable length character string

SQL Type

464, 465

Range

1 ≤ n ≤ 32,767

SQL Type Name

`VARGRAPHIC(n)`

CHAR(n)

Mixed fixed length character string; there are two possibilities, determined by the range necessary for the datatype: converts to `CHAR(n)` for n under 256, and to `VARCHAR(n)` for longer character strings.

Shorter version

SQL Type

452, 453

Range

`1 n 255`

SQL Type Name

`CHAR(n) FOR MIXED DATA`

Longer Version

SQL Type

448, 449

Range

`256 n 32,767`

SQL Type Name

`VARCHAR(n) FOR MIXED DATA`

NCHAR(n)

National fixed length character string; there are two possibilities, determined by the range necessary for the datatype: converts to `CHAR(n)` for n under 256, and to `VARCHAR(n)` for longer character strings.

Shorter version

SQL Type

468, 469

Range

`1 n 255`

SQL Type Name

`GRAPHIC(n)`

Longer Version

SQL Type

464, 465

Range

`256 n 32,767`

SQL Type Name

`VARGRAPHIC(n)`

UROWID

Oracle universal `ROWID`

SQL Type

908, 909

SQL Type Name

`VARBINARY(4000)`

DATE

Oracle `DATE`

SQL Type

384, 385

SQL Type Name

`DATE`

TIMESTAMP

Oracle `TIMESTAMP`

SQL Type

392, 393

SQL Type Name

`TIMESTAMP`

TIMESTAMP WITH LOCAL TIME ZONE

Oracle `TIMESTAMP WITH LOCAL TIME ZONE`

SQL Type

392, 393

SQL Type Name

`TIMESTAMP`

TIMESTAMP(p) WITH TIME ZONE

Oracle `TIMESTAMP WITH LOCAL TIME ZONE`

SQL Type

448, 449

Range

`0 p 9`

SQL Type Name

`VARCHAR(n) FOR MIXED DATA`

`n=148` for `TIMESTAMP(0) WITH TIME ZONE`; otherwise, `149+p` for `TIMESTAMP(p) WITH TIME ZONE`

RAW(n)

Binary variable length string

SQL Type

908, 909

Range

`1 n 2000`

SQL Type Name

`VARBINARY(n)`

NUMBER and FLOAT

Oracle `NUMBER` and `FLOAT` may be used to represent several numeric types:

- simple integer types with only a decimal precision
- fixed-point decimal types with a specific precision and scale
- floating point types with up to `38` decimal digits of precision and an exponent

Additionally, NUMBER may be defined with a scale that is greater than precision, with negative scale, and as a FLOAT with binary precision. See Table 7-1 and Table 7-2 for details.

Note that the general form of this datatype is NUMBER(*p*,*s*), where *p* is the variable for precision and *s* is the variable for scale.

Table 7-1 Converting Oracle NUMBER Variants to DRDA Datatypes

Oracle Variant of NUMBER(p,s)	DRDA Datatype	Notes
NUMBER(1)	DECIMAL(1)	
NUMBER(2-4)	SMALLINT	
NUMBER(5-9)	INTEGER	
NUMBER(10-18)	BIGINT	Whenever the client does not support BIGINT, the mapping is made to DECIMAL(n)
NUMBER(19-31)	DECIMAL(*p*)	
NUMBER(1-31, 1-31)	DECIMAL(p,s)	For both datatypes, scale <= precision
NUMBER(32-38)	DECFLOAT34	Whenever the client does not support DECFLOAT, the mapping is made to DOUBLE. Oracle NUMBER(35-38) is rounded to 34 digits during conversion.
NUMBER(1-38, -s) where scale is negative	DECFLOAT34	Whenever the client does not support DECFLOAT, the mapping is made to DOUBLE. Oracle NUMBER(35-38) is rounded to 34 digits during conversion.
NUMBER(1-38, s) where scale > precision	DECFLOAT34	Whenever the client does not support DECFLOAT, the mapping is made to DOUBLE. Oracle NUMBER(35-38) is rounded to 34 digits during conversion.
NUMBER(32-38, s) with any scale	DECFLOAT34	Whenever the client does not support DECFLOAT, the mapping is made to DOUBLE. Oracle NUMBER(35-38) is rounded to 34 digits during conversion.
NUMBER	DECFLOAT34	Whenever the client does not support DECFLOAT, the mapping is made to DOUBLE

Table 7-2 Converting Oracle FLOAT Variants to DRDA Datatypes

Oracle Variant of FLOAT(n)	DRDA Datatype	Notes
FLOAT(1-53)	DECFLOAT16	Whenever the client does not support DECFLOAT, the mapping is made to DOUBLE
FLOAT(n) where n > 53	DECFLOAT34	Whenever the client does not support DECFLOAT, the mapping is made to DOUBLE
FLOAT	DECFLOAT34	Whenever the client does not support DECFLOAT, the mapping is made to DOUBLE

Datatype Equivalence and Remapping

Oracle does not provide discrete database datatypes such as `SMALLINT`, `INTEGER` or `BIGINT` DRDA datatypes. In some cases, often to limit the column's range of values, it may become necessary to define a numeric column with specific precision or scale. Oracle therefore supplies a more flexible numeric database datatype, Oracle `NUMBER`, which may be defined by specified precision and scale. Oracle `NUMBER` may contain both integral and fractional values in the same column, if no specific range limitations have been defined for the column.

Consideration, therefore, must be made for appropriate database datatypes when migrating data from a non-Oracle database. This is particularly important when migrating applications that expect to handle data of a limited range or form.

For example, if the application accepts a data range specific to `NUMBER(5)`, but the column is defined by datatype `NUMBER`, it is likely that an inappropriate or invalid values may be inserted into the column and causing data issues when using or retrieving that value.

If the table definition is mapped to a close approximation of the original non-Oracle data, there should be no datatype compatibility issues. However, in cases where data that was not modeled accurately must be accessed, or if a query uses an expression that yields a non-range limited datatype, it may become necessary to apply an alternate datatype that is more compatible.

Consider that the `COUNT(*)` expression results in a non-range limited Oracle `NUMBER` datatype. If the application expects the result of the query that uses `COUNT` to be represented as a DRDA `INTEGER` datatype, it becomes necessary perform one of the following steps to avoid a type mis-match:

- change the application to use the Oracle `NUMBER`
- change the query expression to `CAST` the result to the appropriate form
- remap the resulting datatype form

Often, it is neither practical nor feasible to modify the application, and remapping the datatype is the only workable solution.

The Application Server has a limited facility to convert Oracle `NUMBER` datatype results to more discrete equivalent DRDA datatypes, on a per table or per column basis. This mechanism may also be used when the client AR is unable to properly convert the default mappings of Oracle `NUMBER` to DRDA datatype. See “Conversion of Oracle Datatype to DRDA” for all supported conversions.

Related Topics

- Conversion of Oracle Datatype to DRDA

Applying Datatype Mapping

To apply datatype mappings, you must invoke the PL/SQL function. Refer the section on “`SET_TYPEMAP`.” The procedure `SET_TYPEMAP` implements a specified type conversion map for a specified table and column expression. The syntax for the type map object name is *`table_name`*`:`*`column_expression`*. The wildcard character, `*`, may be used in place of table name to include all tables with the specified column

expression. It may also be used to indicate that all column expressions for a specified table that evaluate to an Oracle NUMBER be type mapped.

The syntax for converting from Oracle NUMBER to another datatype is NUMBER=*datatype*. See Oracle NUMBER TYPEMAP Datatype Names for available datatype names.

The default mapping of Oracle NUMBER is to DRDA DECFLOAT(34). “Using TYPEMAP in Queries that Use the Column Directly” shows that queries that use a column directly may use re-mapping on the retrieved column as a DRDA type INTEGER. When using a column in a function it may be necessary to apply a typemap for the expression, as described in “Using TYPEMAP in a function”.

Related Topics

- Using TYPEMAP in Queries

Using TYPEMAP in Queries

Assume that an application expects an EMPLOYEE_ID value to be in a format of DRDA type INTEGER.

Example 7-2 Using TYPEMAP in Queries that Use the Column Directly

```
CREATE TABLE employees(employee_id NUMBER(6), first_name VARCHAR2(20), ...);
```

This mapping enforces range limitations. To facilitate this mapping, apply the following typemap entry for the applications package ORACLE.MYPACKAGE:

```
begin
   dbms_drdaas.set_typemap (
      'ORACLE', 'MYPACKAGE', 'EMPLOYEES:EMPLOYEE_ID',
      'NUMBER=INTEGER');
end;
```

Using TYPEMAP in Functions

When using the COUNT function against the column, as in

Example 7-3 Using TYPEMAP in a Function

```
SELECT COUNT(employee_id) FROM employees;
```

apply the following typemap expression:

```
begin
  dbms_drdaas.set_typemap (
    'ORACLE', 'MYPACKAGE', 'EMPLOYEES:COUNT(EMPLOYEE_ID)',
    'NUMBER=INTEGER' );
end;
```

Oracle NUMBER TYPEMAP

Table 7-3 lists available typemap names and their conversion to DRDA datatypes.

Table 7-3 Oracle NUMBER TYPEMAP Datatype Names

Datatype Name	SQL Type	Datatype Size	Notes
SMALLINT	500, 501	2 bytes	small integer
INTEGER	496, 497	4 bytes	integer
BIGINT	492, 493	8 bytes	large integer
FLOAT	480. 481	4 bytes	single-precision floating point
DOUBLE	480, 481	8 bytes	double-precision floating point

8

Data Dictionary for Oracle Database Provider for DRDA

Oracle enhanced its data dictionary to enable Oracle Database Provider for DRDA.

Data Dictionary Emulation in Oracle Database Provider for DRDA

Many applications use a subset of DB2 system catalogs. Oracle Database Provider for DRDA supports three major sets of catalogs. While all three have some common structures among them, there are many distinct differences.

These catalogs may be implemented as overlays, or views, of existing Oracle data dictionary tables and views.

The catalog described in this section is for “DB2 for z/OS”.

Related Topics

- DB2 for z/OS

DB2 for z/OS

DB2 for z/OS includes the following catalog tables. Please see IBM's DB2 for z/OS SQL Reference manual for description of these views.

- `SYSIBM.SYSCOLUMNS`
- `SYSIBM.SYSDUMMY1`
- `SYSIBM.SYSFOREIGNKEYS`
- `SYSIBM.SYSINDEXES`
- `SYSIBM.SYSKEYCOLUSE`
- `SYSIBM.SYSKEYS`
- `SYSIBM.SYSPACKAGE`
- `SYSIBM.SYSPACKSTMT`
- `SYSIBM.SYSPARMS`
- `SYSIBM.SYSPLAN`
- `SYSIBM.SYSRELS`
- `SYSIBM.SYSROUTINES`
- `SYSIBM.SYSSYNONYMS`
- `SYSIBM.SYSTABCONST`

- SYSIBM.SYSTABLES
- SYSIBM.SYSVIEWS

Data Dictionary Views for Oracle Database Provider for DRDA

Oracle Database Provider for DRDA has uses several data dictionary views:

ALL_DRDAASPACKAGE Data Dictionary View

Table 8-1 contains the list of currently bound DRDA packages. It is owned by user SYSIBM. Users must be granted either the DRDAAS_USER_ROLE or the DRDAAS_ADMIN_ROLE in order to access this view; see "DRDAAS_USER_ROLE" and "DRDAAS_ADMIN_ROLE".

Table 8-1 ALL_DRDAASPACKAGE data dictionary view description

Column Name	Datatype	Null?	Description
COLLID	VARCHAR2(128)	Not NULL	collection ID of Pkg (RDBCOLID)
NAME	VARCHAR2(128)	Not NULL	name of DRDA package (PKGID)
VRSNAM	VARCHAR2(128)		version name of package (VRSNAM)
CONTOKEN	RAW(8)	Not NULL	consistency string of package (PKGCNSTKN)
OWNER	VARCHAR2(128)	Not NULL	userid that owns package
CREATOR	VARCHAR2(128)	Not NULL	userid that created/bound package
CREATE_TIME	TIMESTAMP	Not NULL	time package is created
LAST_BIND_TIME	TIMESTAMP	Not NULL	time of the last binding of package

Related Topics

- DRDAAS_USER_ROLE
- DRDAAS_ADMIN_ROLE

ALL_DRDAASPACKAUTH Data Dictionary View

Table 8-2 contains the set of user ids, DRDA package names, and privileges granted to the user for each package. It is owned by user SYSIBM. Users must be granted either the DRDAAS_ADMIN_ROLE or the DRDAAS_ADMIN_ROLE in order to access this view; see "DRDAAS_USER_ROLE" and "DRDAAS_ADMIN_ROLE".

Table 8-2 ALL_DRDAASPACKAUTH data dictionary view description

Column Name	Datatype	Null?	Description
GRANTOR	VARCHAR2(128)	Not NULL	userid of user who granted privileges
GRANTEE	VARCHAR2(128)	Not NULL	userid of user who received privileges

Table 8-2 (Cont.) ALL_DRDAASPACKAUTH data dictionary view description

Column Name	Datatype	Null?	Description
GRANT_TIME	TIMESTAMP	Not NULL	time privileges were created
COLLID	VARCHAR2(128)	Not NULL	collection ID of DRDA package (RDBCOLID)
NAME	VARCHAR2(128)	Not NULL	name of DRDA package (PKGID)
PRIVILEGE	VARCHAR2(128)	Not NULL	privilege

Related Topics

- DRDAAS_USER_ROLE
- DRDAAS_ADMIN_ROLE

ALL_DRDAASPACKSIDE Data Dictionary View

Table 8-3 shows side attributes for the DRDA package. It is owned by user SYSIBM. Users must be granted either the DRDAAS_ADMIN_ROLE or the DRDAAS_ADMIN_ROLE in order to access this view; refer "DRDAAS_USER_ROLE" and "DRDAAS_ADMIN_ROLE".

Table 8-3 ALL_DRDAASPACKSIDE data dictionary view description

Column Name	Datatype	Null?	Description
COLLID	VARCHAR2(128)	Not NULL	collection ID (Schema) of DRDA package (RDBCOLID)
NAME	VARCHAR2(128)	Not NULL	name of DRDA package (PKGID)
SIDEITEM	VARCHAR2(128)	Not NULL	side item
SIDEWORD	VARCHAR2(255)	Not NULL	side keyword
SIDEVALUE	VARCHAR2(255)	Not NULL	side value

Related Topics

- DRDAAS_ADMIN_ROLE
- DRDAAS_USER_ROLE

DBA_DRDAASPACKAGE Data Dictionary View

Table 8-4 contains the DRDA package definition data. It is owned by user SYSIBM. Users must be granted the DRDAAS_ADMIN_ROLE in order to access this view; see "DRDAAS_ADMIN_ROLE".

Table 8-4 DBA_DRDAASPACKAGE data dictionary view description

Column Name	Datatype	Null?	Description
COLLID	VARCHAR2(128)	Not NULL	collection ID (Schema) of DRDA package (RDBCOLID)

Table 8-4 (Cont.) DBA_DRDAASPACKAGE data dictionary view description

Column Name	Datatype	Null?	Description
NAME	VARCHAR2(128)	Not NULL	name of DRDA package (PKGID)
VRSNAM	VARCHAR2(128)		version name of package (VRSNAM)
CONTOKEN	RAW(8)	Not NULL	consistency string of package (PKGCNSTKN)
OWNER	VARCHAR2(128)	Not NULL	userid that owns package
CREATOR	VARCHAR2(128)	Not NULL	userid that created package
CREATE_TIME	TIMESTAMP	Not NULL	time package is created
LAST_BIND_TIME	TIMESTAMP	Not-NULL	time of the last binding of package
QUALIFIER	VARCHAR2(128)		default schema of package (DFTRDBCOL)
PKSIZE	NUMBER(5)	Not NULL	number of sections in the package (MAXSCTNBR)
VALID	CHAR(1)	Not NULL	package valid state; Y for yes, N for no
ISOLATION	CHAR(1)	Not NULL	R=RR, A=ALL, C=CS, G=CHG, N=NC (PKGISOLVI)
RELEASEOPT	CHAR(1)		C=COMMIT, D=DEALLOCATE (RDBRLSOPT)
BLOCKING	CHAR(1)		B=block, N=no blocking (QRYBLKCTL)
CODEPAGES	NUMBER(5)		default DBCS codepage (PKGDFTCC(CCSIDSBC))
CODEPAGED	NUMBER(5)		Default DBCS codepage (PKGDFTCC(CCSIDDBC))
CODEPAGEM	NUMBER(5)		Default MBCS codepage (PKGDFTCC(CCSIDMBC))
CODEPAGEX	NUMBER(5)		Default XML codepage (PKGDFTCC(CCSIDXML))
DEGREEIOPRL	NUMBER(5)		Degree of parallel I/O (DGRIOPRL)
DATEFMT	CHAR(1)		Date Format, 1=USA, 2=EUR, 3=ISO, 4JIS, 5=LOCAL
TIMEFMT	CHAR(1)		Time Format, 1=USA, 2=EUR, 3=ISO, 4JIS, 5=LOCAL
DECDEL	CHAR(1)		Decimal Delimiter (STTDECDEL)
STRDEL	CHAR(1)		String Delimiter (STTSTRDEL)
DECPRC	NUMBER(5)		Decimal Precision (DECPRC)
CHARSUBTYPE	CHAR(1)		Character Subtype (PKGDFTCST)
DYNAMICRULES	CHAR(1)		Future usage (PKGATHRUL)
REPREPDYNSQL	CHAR(1)		Future usage (PRPSTTKP)

Related Topics

- DRDAAS_ADMIN_ROLE

DBA_DRDAASPACKAUTH Data Dictionary View

Table 8-5 contains the set of userids and RDA package names, as well as privileges granted to the user for each package. It is owned by user SYSIBM. Users must be granted the DRDAAS_ADMIN_ROLE in order to access this view; see "DRDAAS_ADMIN_ROLE".

Table 8-5 DBA_DRDAASPACKAUTH data dictionary view description

Column Name	Datatype	Null?	Description
GRANTOR	VARCHAR2(128)	Not NULL	authorization ID of user who grants package privileges
GRANTEE	VARCHAR2(128)	Not NULL	authorization ID of user who has package privileges
COLLID	VARCHAR2(128)	Not NULL	collection ID (Schema) of DRDA package (RDBCOLID)
NAME	VARCHAR2(128)	Not NULL	name of DRDA package (PKGID)
VRSNAM	VARCHAR(128)	Not NULL	version name of package (VRSNAM)
PRIVILEGE	VARCHAR2(128)	Not NULL	granted privilege

Related Topics

- DRDAAS_ADMIN_ROLE

DBA_DRDAASPACKSIDE Data Dictionary View

Table 8-6 shows side attributes for the DRDA package. It is owned by user SYSIBM. Users must have DRDAAS_ADMIN_ROLE in order to access this view.

Table 8-6 DBA_DRDAASPACKSIDE data dictionary view description

Column Name	Datatype	Null?	Description
COLLID	VARCHAR2(128)	Not NULL	collection ID (Schema) of DRDA package (RDBCOLID)
NAME	VARCHAR2(128)	Not NULL	name of DRDA package (PKGID)
SIDEITEM	VARCHAR(128)	Not NULL	side item
SIDEWORD	VARCHAR2(255)	Not NULL	side keyword
SIDEVALUE	VARCHAR2(255)	Not NULL	side value

DBA_DRDAASPACKSTMT Data Dictionary View

Table 8-7 contains the DRDA package body data. It is owned by user SYS. Users must be granted the DRDAAS_ADMIN_ROLE in order to access this view; see "DRDAAS_ADMIN_ROLE".

Table 8-7 DBA_DRDAASPACKSTMT data dictionary view description

Column Name	Datatype	Null?	Description
COLLID	VARCHAR2(128)	Not NULL	collection ID (Schema) of DRDA package (RDBCOLID)
NAME	VARCHAR2(128)	Not NULL	name of DRDA package (PKGID)
VRSNAM	VARCHAR(128)		version name of package (VRSNAM)
CONTOKEN	RAW(8)	Not NULL	consistency string of package (PKGCNSTKN)
STMTASM	CHAR(1)	Not NULL	Statement Assumptions (BNDSTTADM)
STMTNO	NUMBER(5)	Not NULL	statement number (SQLSTTNBR)
SECTNO	NUMBER(5)	Not NULL	section number (PKGSN)
STMTLEN	NUMBER	Not NULL	statement text length
STMT	CLOB	Not NULL	statement text

Related Topics

- DRDAAS_ADMIN_ROLE

DBA_DRDAASTRACE Data Dictionary View

"DRDAASPATRACE data dictionary view description contains trace entry from the DBMS_DRDAAS and DBMS_DRDAAS_ADMIN package functions. It is owned by user SYSIBM.Users must be granted either the DRDAAS_ADMIN_ROLE or the DRDAAS_ADMIN_ROLE in order to access this view.

This is a debugging feature only.

Table 8-8 DBA_DRDAASPATRACE data dictionary view description

Column Name	Datatype	Null?	Description
CALLER	VARCHAR2(128)	Not NULL	userid of user who wrote the trace record
FUNC	VARCHAR2(128)	Not NULL	function that is traced
ARG1	VARCHAR(128)		argument 2
ARG2	VARCHAR(128)		argument 1
ARG3	VARCHAR(128)		argument 3
ARG4	VARCHAR(128)		argument 4
ARG5	VARCHAR(128)		argument 5
TS	TIMESTAMP	Not-NULL	timestamp of trace entry

Related Topics

- DRDAAS_ADMIN_ROLE
- DRDAAS_USER_ROLE

USER_DRDAASPACKAGE Data Dictionary View

The view only returns rows that match the current SQL user ID in the `creator` or `owner` column and have `EXECUTE_PRIVILEGE` granted to the userid in `DBA_DRDAASPACKAUTH` matching the package `COLLID` and `name`.

The view `USER_DRDAASPACKAGE` maps onto `DBA_DRDAASPACKAGE` table. It is owned by user `SYSIBM`. Users must have the `DRDAAS_USER_ROLE` in order to select from this view; see "`DRDAAS_USER_ROLE`".

This view has the same column definition as the `DBA_DRDAASPACKAGE` table; see "DBA_DRDAASPACKAGE data dictionary view description."

Related Topics

- DBA_DRDAASPACKAGE Data Dictionary View

USER_DRDAASPACKAUTH Data Dictionary View

This view only return rows that match the current SQL user ID in the `grantee` column.

The view `USER_DRDAASPACKAUTH` maps on top of `DBA_DRDAASPACKAUTH`. It is owned by user `SYSIBM`. Users must have the `DRDAAS_USER_ROLE` in order to select from this table; see "`DRDAAS_USER_ROLE`".

This view has the same column definition as the `DBA_DRDAASPACKAUTH` table; see "`DBA_DRDAASPACKAUTH` data dictionary view description".

Related Topics

- DRDAAS_USER_ROLE
- DBA_DRDAASPACKAUTH Data Dictionary View

USER_DRDAASPACKSIDE Data Dictionary View

This view is used internally by the Application Server.

The view `USER_DRDAASPACKSIDE` maps on top of `DBA_DDRDAASPACKSIDE`. It is owned by user `SYSIBM`. Users must have the `DRDAAS_USER_ROLE` in order to select from this table; see "`DRDAAS_USER_ROLE`".

This view has the same column definition as the `DBA_DRDAASPACKSIDE` table; see "`DBA_DRDAASPACKSIDE` data dictionary view description".

Related Topics

- DRDAAS_USER_ROLE
- DBA_DRDAASPACKSIDE Data Dictionary View

USER_DRDAASPACKSTMT Data Dictionary View

The view only returns rows that match the current SQL user ID in the `creator` or `owner` column, and have `EXECUTE_PRIVILEGE` granted to the userid in `DBA_DRDAASPACKAUTH` matching the package `COLLID` and `name`.

The view USER_DRDAASPACKSTMT maps onto of DBA_DRDAASPACKSTMT table. It is owned by user SYSIBM. Users must have the DRDAAS_USER_ROLE in order to select from this table; see "DRDAAS_USER_ROLE".

This view has the same column definition as the DBA_DRDAASPACKSTMT table; see "DBA_DRDAASPACKSTMT data dictionary view description."

Related Topics

- DRDAAS_USER_ROLE
- DBA_DRDAASPACKSTMT Data Dictionary View

USER_DRDAASTRACE Data Dictionary View

The view returns only the rows that match the userid of the current user.

This is a debugging feature only.

The view USER_DRDAASTRACE maps onto DBA_DRDAASTRACE view. It is owned by user SYSIBM. Users must be granted the DRDAAS_USER_ROLE in order to access this view; see "DRDAAS_USER_ROLE".

This view has the same column definition as the USER_DRDAASTRACE view, with the exception of the caller column. See "DBA_DRDAASPATRACE data dictionary view".

Related Topics

- DRDAAS_USER_ROLE
- DBA_DRDAASTRACE Data Dictionary View

9

Error Codes Support in Oracle Database Provider for DRDA

Oracle Database Provider for DRDA has error code support, and supports management of errors and warnings generated by applications developed for DB2 and DRDA.

Oracle Error Codes

In DRDA and DB2, certain types of operations may be successful but generate a warning. For example, when an operation generates an integer overflow situation, DB2 continues and returns the rest of the row data with an indicator set for the column value that contains the error. It will also issue the warning code `+802`:

```
(EXCEPTION ERROR exceptioncode HAS OCCURRED DURING
   operationtype OPERATION
   ON datatype DATA,
   POSITION positionnumber)
```

In DB2 and DRDA, negative `SQLCODE`s are errors, and positive `SQLCODE`s are warnings. Oracle, unlike DRDA (and DB2), has no notion of a warning condition. In Oracle, call execution is either successful or generates an error. For example, Oracle treats an overflow situation as an error, and does not continue. Thus, Oracle Database Provider for DRDA cannot emulate DB2 behavior precisely.

However, most normal error conditions that are common to both Oracle and DRDA (and DB2) may be mapped.

“Error Code Mappings, from Oracle to DRDA” lists some common Oracle error codes and their equivalent SQL codes and SQL states.

Related Topics

- Error Code Mapping, from Oracle to DRDA

Error Code Mapping, from Oracle to DRDA

Table 9-1 Error Code Mappings, from Oracle to DRDA

Oracle Error Code	SQLCODE	SQLSTATE	Error Code Description
ORA-00001	-803	23505	Unique constraint violation
ORA-00900	-104 or -199	42601	Invalid SQL statement
ORA-00900	-84	42616	Invalid SQL statement
ORA-00901	-104 or -199	42601	Invalid CREATE command
ORA-00902	-104 or -199	42601	Invalid datatype

Table 9-1 (Cont.) Error Code Mappings, from Oracle to DRDA

Oracle Error Code	SQLCODE	SQLSTATE	Error Code Description
ORA-00903	-104 or -199	42601	Invalid table name
ORA-00904	-204	42704	Invalid identifier
ORA-00904	-206	42703	Invalid identifier
ORA-00905	-199	42601	Misspelled keyword
ORA-00906	-104	42601	Missing left parenthesis
ORA-00907	-104	42601	Missing right parenthesis
ORA-00908	-104	42601	Missing NULL
ORA-00909	-170	42605	Incorrect number of arguments
ORA-00910	-604	42611	Creating a CHAR column with length of max +1
ORA-00911	-104 or -199	42601	Invalid character
ORA-00913	-117	42802	Too many values
ORA-00917	-104 or -199	42601	Missing coma
ORA-00918	-203	42702	Ambiguous column usage
ORA-00920	-104 or -199	42601	Invalid relational operator
ORA-00922	-104 or -199	42601	Invalid or missing option
ORA-00923	-104	42601	FROM keyword not found where expected
ORA-00925	-104	42601	Missing INTO keyword
ORA-00926	-104	42601	Missing VALUES keyword
ORA-00927	-199	42601	Missing = sign
ORA-00928	-104	42601	Missing SELECT keyword
ORA-00932	-408	42821	Value not compatible with target column type
ORA-00933	-104 or -199	42601	SQL command not properly ended
ORA-00934	-120	42903	Group function not allowed
ORA-00936	-104 or -199	42601	Missing expression
ORA-00937	-122	42803	Not a single group function
ORA-00942	-204	42704	Table or view does not exist
ORA-00947	-117	42802	Not enough values
ORA-00950	-104 or -199	42601	Invalid DROP option
ORA-00955	-601	42710	Name is already used by an existing object
ORA-00957	-612	42711	Duplicate column name
ORA-00960	-203	42702	Ambiguous column naming in select list
ORA-00969	-104	42601	Missing ON keyword
ORA-00971	-104	42601	Missing SET keyword

Table 9-1 (Cont.) Error Code Mappings, from Oracle to DRDA

Oracle Error Code	SQLCODE	SQLSTATE	Error Code Description
ORA-00972	-107	46002	Identifier is too long
ORA-00975	-182	42816	Date + date not allowed
ORA-00979	-122	42803	Not a GROUP BY expression
ORA-00999	-104 or -199	42601	Invalid view name
ORA-01000	-905	57014	Maximum open cursors exceeded
ORA-01002	-501	22003	Fetch out of sequence
ORA-01008	-313	07001	Not all variables bound
ORA-01031	-551	42501	Insufficient privilege to perform operation on object
ORA-01036	-313	07001	Illegal variable name/number
ORA-01400	-407	23502	Cannot insert NULL into column
ORA-01403	+100	02000	No data found
ORA-01410	-399	22511	Invalid ROWID
ORA-01422	-811	21000	Exact fetch returns more than requested number of rows
ORA-01424	-130	22019	Missing or illegal character following the escape character
ORA-01425	-130	22025	Escape character must be string of length 1
ORA-01427	-811	21000	Single-row subquery returns more than one row
ORA-01435	-553	42503	User does not exist
ORA-01438	-413	22003	Value larger than specified precision allowed for this column
ORA-01455	-413	22003	Converting column overflows INTEGER datatype
ORA-01476	-802	22012	Divisor equal to zero
ORA-01488	-302	22001	Invalid nibble or byte in the input data
ORA-01722	-408	42821	Invalid number
ORA-01730	-158	42811	CREATE VIEW with ambiguous number of columns
ORA-01745	-313	07001	Invalid host/bind variable name
ORA-01747	-104 or -199	42601	Invalid table or column specification
ORA-01756	-104	42603	Missing quote for quoted string
ORA-01789	-421	42826	Operands of a set operator do not have the same number of columns
ORA-01830	-181	22007	Date format picture ends before converting entire input string

Table 9-1 (Cont.) Error Code Mappings, from Oracle to DRDA

Oracle Error Code	SQLCODE	SQLSTATE	Error Code Description
ORA-01841	-181	22007	(Full) year must be between -4713 and +9999, and not 0
ORA-01843	-181	22007	Bad month
ORA-01847	-181	22007	Bad year
ORA-01858	-181	22007	not numeric where numeric expected in date
ORA-01861	-180	22007	Literal does not match format string
ORA-02089	-426	2D528	COMMIT is not allowed in subordinate session
ORA-02291	-530	23503	Parent not found (insert/update/delete)
ORA-02292	-531	23504	Child not found (insert/update)
ORA-02292	-532	23504	Child not found (delete)
ORA-04043	-204	42704	Object not found
ORA-04063	-84	42612	Package body has errors
ORA-06550	-204	42504	Reports various PL/SQL errors, such as: • PLS00201: identifier must be declared • PLS-00904: insufficient privilege to access object
ORA-06576	-440	42884	Not a valid function or procedure name
ORA-08006	-508	24504	Specified row no longer exists
ORA-12899	-404	22001	Value too large for column
ORA-20980	-551	42501	User does not have package privilege for operation
ORA-20981	-917	42969	Bind package failed
ORA-20982	-551	42501	User does not have package privilege for operation
ORA-20983	-722	42704	Package does not exist
ORA-22275	-423	0F001	Invalid LOB locator specified
ORA-24333	-104 or -199	42601	Misspelled SQL statement
ORA-24381	-253	22529	array insert or merge reported some errors
All other errors	-84	42612	

10

Command-line Utility for Oracle Database Provider for DRDA

Oracle Database Provider for DRDA provides command-line utility.

Command-line Utility

The command-line utility `drdactl` enables the user to control the Oracle DRDA Application Sever. It controls startup, shutdown, status and operational changes of the AS. The `drdactl` utility may be invoked with command arguments for immediate execution, or without arguments (the utility will prompt for commands).

START

This command starts the designated instance.

Syntax

```
START {instance_name}
```

STOP

This command stops the designated running instance.

Syntax

```
STOP {instance_name}
```

STATUS

This command displays the current status of the designated running instance, and any currently connected session information.

Syntax

```
STATUS {instance_name} {DETAIL}
```

TRACE

This command disables or enables tracing at a specified level, for the specified session id.

Syntax

```
TRACE {<instance_name>} <OFF|Level> {SESSION {<session_id>|ALL}}
```

PAUSE

This command pauses the session with the specified session id.

Syntax

```
PAUSE {instance_name} SESSION <session_id>
```

RESUME

This command resumes the session with the specified session id.

Syntax

```
RESUME {instance_name} SESSION <session_id>
```

RELOAD

This command causes the server to reload the instance configuration.

Syntax

```
RELOAD {instance_name}
```

EXIT

This command exits the utility and returns to the operating system.

Syntax

```
EXIT
```

11

Security and Storage Considerations for Oracle Database Provider for DRDA

Before using Oracle Database Provider for DRDA to transition DB2 DRDA applications to Oracle, several security and storage issues must be considered.

Overview of Security and Storage for Oracle Database Provider for DRDA

Oracle Database Provider for DRDA uses the security and authorization models of the Oracle Database; this ensures correct SQL access to user data.

DRDA specifies two primary authorization models: `REQUESTER` and `OWNER`. They are specified as attribute of the package, `PKGATHRUL`.

- In the `REQUESTER` model, currently logged-in user is the authorization control.

  The `REQUESTER` model is also called Dynamic SQL Rules, and it is used for executing the SQL that is constructed at runtime. This is the security model that Oracle Database implements for SQL access.

- In the `OWNER` model, a stored `USERID` (package owner, `PKGOWNID`) instantiates a different `USERID` for the duration of the statement execution.

  The `OWNER` model is also called Static SQL Rules, and it is used for pre-bound SQL within the package. Oracle Database does not implement an equivalent security model.

Because Oracle Database only implements one security model, all SQL within a package, both stored and dynamically created, is executed under the Dynamic SQL Rules model. This does not override the instantiation modes of any PL/SQL objects created with the `AUTHID DEFINER` attribute.

See *Oracle® Database PL/SQL Language Reference* for details about `AUTHID DEFINER`.

Authentication and Encryption in Oracle Database Provider for DRDA

DRDA requires that user authentication (ACCSEC(SECMEC), SECCHK) be performed before any access to the database (ACCRDB) may be performed.

Authentication Services

DRDA provides two types of services for authentication: dedicated use, and multiplexed use.

- Under **dedicated** use, the AS provides access to a single, dedicated database, without an option to connect to another database. The rdbnam passed later in the ACCRDB must match the target database name; otherwise, the session access fails with an RDBAFLRM message.

  In this mode, the ACCSEC does not require the rdbnam instance variable to perform authentication because there is no choice to be made between databases. Thus, the AS is free to make a session association with the single database at the time of ACCSEC processing, and to verify that the requested security mechanism (SECMEC) is available. When the SECCHK command is sent, the initial database session is maintained and full authentication is performed.

  This mode must be used with older ARs that do not support multiplexed servers. If the AR supplies the rdbnam in this mode, it must be validated against the connected database.

- Under **multiplexed** mode, the AS provides access to many databases at the same time. In this mode, the rdbnam instance variable in the ACCSEC is required to validate the requested security mechanism. The AS connects to the requested database for security validation. When the SECCHK command is sent, the initial database session is maintained and full authentication is performed.

  This mode is supported with newer ARs, which supply the rdbnam as part of the ACCSEC. If the AR does not supply the rdbnam in the ACCSEC in this mode, it generates an error and an RDBAFLRM response is sent to the ACCSEC.

Encryption Services

Oracle Database Provider for DRDA supports the following security mechanisms:

- **EUSRIDPWD** – Encrypted User ID and Password Mechanism
- **EUSRIDNWPWD** – Encrypted User ID, Password, and New Password Mechanism
- **USRIDPWD** – User ID and Password Mechanism
- **USRIDNWPWD** – User ID, Password, and New Password Mechanism
- **USRENCPWD** – User ID and Password Encrypted Mechanism

After the security mechanisms are verified as accessible, the actual security authorization (SECCHK) can proceed.

Database Roles in Oracle Database Provider for DRDA

Oracle Database Provider for DRDA uses the following two roles: `DRDAAS_ADMIN_ROLE` and `DRDAAS_USER_ROLE`.

DRDAAS_ADMIN_ROLE

The `DRDAAS_ADMIN_ROLE` role allows access to and use of the `DBMS_DRDAAS_ADMIN` package. It is a DBA-level privilege specific to Oracle Database Provider for DRDA. Administrators who manage remote DRDA access to the Oracle database must be granted this role.

DRDAAS_USER_ROLE

The DRDAAS_USER_ROLE role permits an Oracle Database Provider for DRDA user access to the DBMS_DRDAAS package for the following purposes:

- For binding a DRDA package
- For permitting read access to Oracle Database Provider for DRDA package resource tables
- For package execution

Without the privileges of the DRDAAS_USER_ROLE role, Oracle Database Provider for DRDA user is unable to execute DRDA packages to which they have granted access.

Storage in Oracle Database Provider for DRDA

Oracle Database Provider for DRDA users have allocated storage named SYSIBM. This is implemented by the SYSIBM tablespace and SYSIBM user.

SYSIBM Tablespace

All tables, views and packages supplied and used by the Application Server for management and support functions are in the tablespace SYSIBM. This tablespace must be created before installing Oracle Database Provider for DRDA support packages. An example of how to create the SYSIBM tablespace is the catdrdaas.sql script supplied with the product. See the listing for “catdrdaas.sql” in “Scripts for Creating and Maintaining Oracle Database Provider for DRDA”.

```
create tablespace SYSIBM datafile 'sysibm01.dbf' size 70M reuse
   extent management local segment space management auto online;
```

Related Topics

- catdrdaas.sql
- Scripts for Creating and Maintaining Oracle Database Provider for DRDA

SYSIBM User

All tables, views, and packages supplied and used by Oracle Database Provider for DRDA are under the user schema SYSIBM. As part of the installation of Oracle Database Provider for DRDA packages and tables, this user id is created as a locked account, and is set to use the SYSIBM tablespace for its storage.

12

Restrictions on Using Oracle Database Provider for DRDA

Several restrictions and known workarounds may be used when customizing or maintaining in Oracle Database applications that were originally designed for IBM DB2.

Resynch Manager

Oracle Database Provider for DRDA supports Sync Point Manager services for Distributed Units of Work. It also supports Resynchronization Manager services for resynchronization during migrations that use a source Sync Point Manager without a log.

This release of Oracle Database Provider for DRDA does not support active in-doubt transaction resolution services. Transactions that have been migrated and are in-doubt require manual resolution between the client system and Oracle Database. See *Oracle® Database Administrator's Guide* for information on manual resolution of in-doubt transactions.

Cursor HOLD Attribute Semantics

Cursors marked with the `HOLD` attribute have the following restrictions:

1. Under Remote Unit of Work (RUOW), cursors that have been prepared with the `FOR UPDATE` clause are implicitly closed on `COMMIT` or `ROLLBACK`.
2. Under Distribute Unit of Work (DUOW), all cursors are implicitly closed if any updates occur to the server containing the open cursor on `COMMIT` or `ROLLBACK`.

DB2 Password Blank Padding

When passwords are encrypted and sent through DRDA, DB2 for z/OS inserts blank spaces into passwords that have less than 8 characters. This results in a log-on failure, error `ORA-01017`. Oracle recommends that user account passwords be at least `8` characters long.

Restrictions on Datatypes

There are several restrictions on use of datatypes.

DATE Datatype

Oracle `DATE` datatype contains a time component that DRDA `DATE` datatype does not support. Operating on Oracle `DATE` data may not yield expected results if the `DATE` data contains a time component. For consistency, do not store a time component when

inserting DATE data using Oracle native DATE syntax. Alternatively, remap the DATE column to TIMESTAMP.

Oracle Object-Relational Datatypes

This release does not support queries on objects that contain columns defined through Object-Relational datatypes.

This release does not support calling SQL procedures defined through Object-Relational datatypes for their input or return arguments.

TIMESTAMP Datatype

Oracle Database Provider for DRDA represents TIMESTAMP with a fixed precision of 6 decimal places.

For compatibility reasons, extra care should be exercised when using TIMESTAMP data, and programmatic adjustments, such as type casting, may have to be made. See *Oracle® Database SQL Language Reference* for information about casting with the TIMESTAMP datatype.

TIMESTAMP WITH TIMEZONE Datatype

Representation of TIMESTAMP WITH TIMEZONE is significantly different between Oracle Database and DB2.

Oracle Database Provider for DRDA represents TIMESTAMP WITH TIMEZONE according to Oracle's presentation rules. For best compatibility between client and server, use four digit time zone suffix notation instead of written timezone description notation, such as -08:00.

XML Datatype

The DRDA XML datatype (988, 989) is not supported as a program or bind variable datatype in this release.

SYS.XMLType Datatype

The Oracle XML datatype, SYS.XMLType, is not supported in this release.

Extended Length Mode

The latest release of DB2 for z/OS (v11.1) does not support ‘Streaming Layer B mode’ protocol for query results, generated by cursors. These cursors are defined using ROWSETs parameter, that generate a rowset exceeding 32767 bytes of data.

Example of cursor declaration

```
EXEC SQL DECLARE DT CURSOR WITH ROWSET POSITIONING FOR SEL;
```

According to the DRDA standard, the target server can determine the returned form of query data for any given query. Streaming Layer B mode is the most efficient form, for this purpose. However, some releases of DB2 do not support this more when

ROWSETs are involved and will cause an error to be returned to the DB2 client application:

```
DSNT4081 SQLCODE = –30020, ERROR: EXECUTION FAILED DUE TO A DISTRIBUTION
PROTOCOL ERROR THAT CAUSED DEALLOCATION OF THE CONVERSATION: REASON 124C
(0100)
```

Under these conditions, DB2 will not accept Streaming Layer B mode objects. However, it will accept Extended Length mode objects instead.

You can set the `PROTOPROC_OPTIONS` configuration parameter to enable this mode until DB2 supports Streaming Layer B mode.

Related Topics

- PROTOPROC_OPTIONS

> **See Also:**
>
> `PROTOPROC_OPTIONS` Section, in the Configurations Parameters Chapter.

DB2 for z/OS Log usage

LOGNAME

The `LOGNAME` value specifies a fixed Log name exchanged between the server and client. The Log Name should be in the format `<NAME>`. The Log name can contain between 1 and 18 alpha-numeric characters only. In case there is an invalid value or length entered, it will be rejected, and a random name will be generated.

LOGTSTMP

The `LOGTSTMP` specifies a fixed Log Timestamp exchanged between the server and client. The Log Timestamp should be in the format `<YYYYMMDDHHMMSSTTTT>` and have only numeric values. In case an invalid value or length is entered, it will be rejected and a timestamp with the current time will be generated.

Following is an illustration:

```
drdaas.protoproc_options="QRYDTA/ELMODE, EXTDTA/ELMODE, LOGNAME/ORACLEDB,
LOGTSTMP/20160919120102000l"
```

Other Restrictions

Other restrictions, such as “SQL Clause Restrictions” are outlined in “SQL Statement Support in Oracle Database Provider for DRDA.”

Related Topics

- SQL Clause Restrictions
- SQL Statement Support in Oracle Database Provider for DRDA

13

PL/SQL Packages Used by Oracle Database Provider for DRDA

Oracle Database Provider for DRDA uses `DBMS_DRDAAS_ADMIN` and `DBMS_DRDAAS` PL/SQL packages and their APIs.

For in-depth information on the type map values used in these two packages, see Datatype Support and Conversion in Oracle Database Provider for DRDA .

DBMS_DRDAAS_ADMIN Package

`DBMS_DRDAAS_ADMIN` PL/SQL package grants DRDA package privileges to Oracle Database Provider for DRDA users. These privileges include the following:

- bind DRDA packages
- drop DRDA packages
- execute DRDA packages
- set package values

DBMS_DRDAAS_ADMIN Privilege Constants

These constants are used with "`GRANT_PRIVILEGE`" and "`REVOKE_PRIVILEGE`".

ALL_PRIVILEGE

This privilege grants all privileges to a client for an Application Package.

BIND_PRIVILEGE

This privilege allows a client to bind or rebind an Application Package to the database.

COPY_PRIVILEGE

This privilege allows a client to copy an existing Application Package to another name (optionally with different default package options).

EXECUTE_PRIVILEGE

This privilege allows a client to execute an existing Application Package.

DROP_PRIVILEGE

This privilege allows a client to drop an existing Application Package.

SET_PRIVILEGE

This privilege allows a client to set specific Application Package options. See the `SET_`*`XXX`* functions elsewhere in this document.

Related Topics

- GRANT_PRIVILEGE
- REVOKE_PRIVILEGE

GRANT_PRIVILEGE

Grants a privilege to the user for a DRDA package.

Syntax

```
PROCEDURE grant_privilege(
   privilege_grant IN PLS_INTEGER,
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   user_name IN VARCHAR2);
```

Parameters

- `privilege_grant (IN)`

  Privilege to grant
- `collection_id (IN)`

  Collection Id
- `package_name (IN)`

  Package Name
- `user_name (IN)`

  Userid to grant privileges to

Usage Example

```
begin
   dbms_drdaas_admin.grant_privilege ( DBMS_DRDAAS_ADMIN.ALL_PRIVILEGE, 'ORACLE',
     'MYPACKAGE', 'DRDAUSR1' );
end;
```

REVOKE_PRIVILEGE

Revokes a privilege from a user for a DRDA package.

Syntax

```
PROCEDURE revoke_privilege(
   privilege_revoke IN PLS_INTEGER,
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   user_name IN VARCHAR2);
```

Parameters

- `privilege_revoke (IN)`

  Privilege to revoke
- `collection_id (IN)`

Collection Id

- `package_name (IN)`

  Package Name

- `user_name (IN)`

  Userid to revoke privileges from

Usage Example

```
begin
   dbms_drdaas_admin.revoke_privilege ( DBMS_DRDAAS_ADMIN.ALL_PRIVILEGE, 'ORACLE',
     'MYPACKAGE', 'DRDAUSR1' );
end;
```

DROP_PACKAGE

Drops all instances of a package by `package_name`.

Syntax

```
procedure DROP_PACKAGE(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2 );
```

Parameters

- `collection_id (IN)`

  Collection Id

- `package_name (IN)`

  Package Name

Usage Example

```
begin
  dbms_drdaas_admin.drop_package(
    'ORACLE', 'MYPACKAGE' );
end;
```

DROP_PACKAGE_VN

Drops a package by `version_name`.

Syntax

```
procedure DROP_PACKAGE_VN(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   version_name IN VARCHAR2 DEFAULT NULL );
```

Parameters

- `collection_id (IN)`

  Collection Id

- `package_name (IN)`

Package name

- `version_name (IN)`

  Version name

DROP_PACKAGE_CT

Drops a package by `consistency_token`.

Syntax

```
procedure DROP_PACKAGE_CT(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   consistency_token IN RAW );
```

Parameters

- `collection_id (IN)`

  Collection Id

- `package_name (IN)`

  Package name

- `consistency_token (IN)`

  Consistency token

SET_PROFILE

Sets the SQL Translation profile name for a DRDA package.

Syntax

```
PROCEDURE set_profile(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   profile_name IN VARCHAR2);
```

Parameters

- `collection_id (IN)`

  Collection Id

- `package_name (IN)`

  Package Name

- `profile_name (IN)`

  SQL Translation profile name

Usage Example

```
begin
   dbms_drdaas_admin.set_profile ( 'ORACLE', 'MYPACKAGE', 'DB2ZOS');
end;
```

SET_LOCALDATE_FORMAT

Sets the Local Date Format to use with a DRDA package.

Syntax

```
PROCEDURE set_localdate_format(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   date_format IN VARCHAR2);
```

Parameters

- `collection_id (IN)`

  Collection Id

- `package_name (IN)`

  Package Name

- `date_format (IN)`

  date format string

Usage Example

```
begin
   dbms_drdaas_admin.set_localdate_format ( 'ORACLE', 'MYPACKAGE', 'YYYYMMDD');
end;
```

SET_LOCALTIME_FORMAT

Sets the local time format to use with a DRDA package.

Syntax

```
PROCEDURE set_localtime_format(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   time_format IN VARCHAR2);
```

Parameters

- `collection_id (IN)`

  Collection Id

- `package_name (IN)`

  Package Name

- `time_format (IN)`

  time format String

Usage Example

```
begin
   dbms_drdaas_admin.set_localtime_format ( 'ORACLE', 'MYPACKAGE', 'HH:MM:SS');
end;
```

SET_TYPEMAP

Sets datatype mapping rules for specific table and column combinations.

Syntax

```
PROCEDURE set_typemap(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   table_map IN VARCHAR2,
   type_map IN VARCHAR2);
```

Parameters

- `collection_id (IN)`

  Collection Id

- `package_name (IN)`

  Package Name

- `table_map (IN)`

  table and column name expression

- `type_map (IN)`

  numeric type equivalence expression

Usage Example

```
begin
   dbms_drdaas_admin.set_typemap ( 'ORACLE', 'MYPACKAGE',
      'SYSIBM.SYSPACKSTMT:COUNT(DISTINCT(NAME))', 'NUMBER=INTEGER');
end;
```

DBMS_DRDAAS Package

`DBMS_DRDAAS` PL/SQL package manipulates DRDA packages. Use this package to bind new DRDA packages, modify attributes of existing DRDA packages, or drop DRDA packages.

Oracle Database Provider for DRDA uses package `DBMS_DRDAAS` to perform specific DRDA package operations.

DBMS_DRDAAS Privilege Constants

These constants are used with “`GRANT_PRIVILEGE`” and “`REVOKE_PRIVILEGES`” procedures.

ALL_PRIVILEGE

This privilege grants all of the above privileges to a client for an Application Package.

BIND_PRIVILEGE

This privilege allows a client to bind or rebind an Application Package to the database.

COPY_PRIVILEGE

This privilege allows a client to copy an existing Application Package to another name (optionally with different default package options).

EXECUTE_PRIVILEGE

This privilege allows a client to execute an existing Application Package.

DROP_PRIVILEGE

This privilege allows a client to drop an existing Application Package.

SET_PRIVILEGE

This privilege allows a client to set specific Application Package options. See the `SET_XXX` functions elsewhere in this document.

Related Topics

- GRANT_PRIVILEGE
- REVOKE_PRIVILEGE

BIND_PACKAGE

Creates the beginnings of a DRDA package definition.

This is used internally by Oracle Database Provider for DRDA part of `BGNBND` processing.

Syntax

```
PROCEDURE bind_package(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   version_name IN VARCHAR2 DEFAULT NULL,
   consistency_token IN RAW,
   owner IN VARCHAR2,
   qualifier IN VARCHAR2,
   isolation IN CHAR,
   releaseopt IN CHAR,
   blocking IN CHAR DEFAULT 'N',
   codepage_s IN NUMBER,
   codepage_d IN NUMBER,
   codepage_m IN NUMBER,
   codepage_x IN NUMBER,
   degreeioprl IN NUMBER,
   date_format IN CHAR DEFAULT '3',
   time_format IN CHAR DEFAULT '3',
   decimal_delimiter IN CHAR DEFAULT NULL,
   string_delimiter IN CHAR DEFAULT NULL,
   decprc IN NUMBER,
   charsubtype IN CHAR,
   dynamic_rules IN CHAR DEFAULT NULL,
   reprepdynsql IN CHAR DEFAULT NULL );
```

Parameters

- `collection_id (IN)` is collection ID

- `package_name (IN)` is package name
- `version_name (IN)` is version name (optional, default `NULL`)
- `consistency_token (IN)` is consistency token
- `owner (IN)` is owner of package
- `qualifier (IN)` is default schema
- `isolation (IN)` is isolation level (`R=RR`, `A=ALL`, `C=CS`, `G=CHG`, `N=NC`)
- `releaseopt (IN)` is release package resource option
- `blocking (IN)` is blocking mode (`B`=blocking, `N`=no blocking)
- `codepage_s (IN)` is default codepage (SBCS)
- `codepage_d (IN)` is default codepage (DBCS)
- `codepage_m (IN)` is default codepage (MBCS)
- `codepage_x (IN)` is default codepage (XML)
- `degreeioprl (IN)` is degree of IO parallelism
- `date_format (IN)` is date format (`1`=USA, `2`=EUR, `3`=ISO, `4`=JIS, `5`=Local)
- `time_format (IN)` is time format (`1`=USA, `2`=EUR, `3`=ISO, `4`=JIS, `5`=Local)
- `decimal_delimiter (IN)` is decimal delimiter
- `string_delimiter (IN)` is string delimiter
- `decprc (IN)` is the decimal precision (15 or 31)
- `charsubtype (IN)` is character subtype
- `dynamic_rules (IN)` is dynamic rules (future)
- `reprepdynsql (IN)` is prepare dynamic SQL rules again (future)

Usage Example

```
begin
   dbms_drdaas.bind_package (
      'ORACLE', 'MYPACKAGE', NULL, HEXTORAW('11223344'), 'DRADUSR1',
      'PETER', 'C', 'D', 'B', 1208, 1200, 1208, 1208, 1, '3', '3', '.', '''',
      31, 'M', 'R', 'Y' );
end;
```

BIND_STATEMENT

Inserts a statement into DRDA package currently being bound.

This is used internally by Oracle Database Provider for DRDA as part of `BNDSQLSTT` processing.

Syntax

```
PROCEDURE bind_statement(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   version_name IN VARCHAR2 DEFAULT NULL,
   consistency_token IN RAW,
   statement_assumption IN CHAR,
```

```
   statement_no IN NUMBER,
   section_no IN NUMBER,
   statement_len IN NUMBER,
   statement IN CLOB );
```

Parameters

- `collection_id (IN)` is collection Id
- `package_name (IN)` is package name
- `version_name (IN)` is version name (optional, default `NULL`)
- `consistency_token (IN)` is consistency token
- `statement_assumption (IN)` is statement assumption
- `statement_no (IN)` is statement number
- `section_no (IN)` is section number
- `statement_len (IN)` is length of SQL statement text
- `statement (IN)` is statement text

Usage Example

```
begin
   dbms_drdaas.bind_statement ( 'ORACLE', 'MYPACKAGE', NULL, HEXTORAW('11223344'),
      'C', 1, 1, 42, 'DECLARE CURSOR C1 AS SELECT EMPLOYEE_ID FROM EMPLOYEES' );
end;
```

END_BIND

Finalizes a DRDA package currently being bound. (This is used internally by Oracle Database Provider for DRDA as part of `ENDBND` processing.)

Syntax

```
PROCEDURE end_bind(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   version_name IN VARCHAR2 DEFAULT NULL,
   consistency_token IN RAW,
   max_sections IN NUMBER );
```

Parameters

- `collection_id (IN)` is collection ID
- `package_name (IN)` is package name
- `version_name (IN)` is version name (optional, default `NULL`)
- `consistency_token (IN)` is the consistency token
- `max_sections (IN)` is the maximum number of sections

Usage Example

```
begin
   dbms_drdaas.end_bind ( 'ORACLE', 'MYPACKAGE', NULL,
      HEXTORAW('11223344'), 1 );
end;
```

GRANT_PRIVILEGE

Grants a privilege on a package to a user.

Syntax

```
PROCEDURE grant_privilege(
   privilege_grant IN PLS_INTEGER,
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   user_name IN VARCHAR2 );
```

Parameters

- `privilege_grant (IN)`

  Privilege to grant
- `collection_id (IN)`

  Collection Id
- `package_name (IN)`

  Package Name
- `user_name (IN)`

  Userid to grant privileges to

Usage Example

```
begin
   dbms_drdaas.grant_privilege ( DBMS_DRDAAS_ADMIN.ALL_PRIVILEGE, 'ORACLE',
     'MYPACKAGE', 'DRDAUSR1' );
end;
```

REVOKE_PRIVILEGE

Revokes a privilege from a user for a DRDA package.

Syntax

```
PROCEDURE revoke_privilege(
   privilege_revoke IN PLS_INTEGER,
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2,
   user_name IN VARCHAR2);
```

Parameters

- `privilege_revoke (IN)`

  Privilege to revoke
- `collection_id (IN)`

  Collection Id
- `package_name (IN)`

  Package Name

- `user_name (IN)`

  Userid to revoke privileges from

Usage Example

```
begin
   dbms_drdaas.revoke_privilege ( DBMS_DRDAAS_ADMIN.ALL_PRIVILEGE, 'ORACLE',
     'MYPACKAGE', 'DRDAUSR1' );
end;
```

DROP_PACKAGE

Drops a DRDA package using the version name.

Syntax

```
PROCEDURE drop_package(
   collection_id IN VARCHAR2,
   package_name IN VARCHAR2);
```

Parameters

- `collection_id (IN)` is the collection id
- `package_name (IN)` is package name

Usage Example

```
begin
  dbms_drdaas.drop_package(
    'ORACLE', 'MYPACKAGE');
end;
```

14

SQL Statement Support in Oracle Database Provider for DRDA

Oracle Database Provider for DRDA supports the use of native DB2 SQL clauses.

Overview of SQL Statement Support

Oracle Database Provider for DRDA transforms parts of the third-party native SQL statements before sending them for processing on the Oracle Database. In this release, Oracle Database Provider for DRDA is made compatible with Oracle Database Release 11*g*, which does not have a native understanding of many clauses not supported by Oracle's version of SQL because it does not support SQL Translation. When using SQL Translation, this affects the data and content of SQL requests received by the translator.

Because the Translator never gets the `WITH UR` clause, the translation of the statement and the subsequent result set may not be what the user expects.

For this reason, this release of Oracle Database Provider for DRDA encompasses some translations functions.

Example 14-1 Removing Clauses from SQL Statements

If a user enters the following SQL line on the client:

```
SELECT * FROM EMPLOYEES WITH UR
```

Oracle Database Provider for DRDA strips out the clause `WITH UR`, so if the user is using a Translator, it receives the following line of SQL:

```
SELECT * FROM EMPLOYEEES
```

SQL Clause Restrictions

There are some restrictions on SQL statements that are supported for conversion by Oracle Database Provider for DRDA.

When describing SQL statements, keep in mind the following notation.

- Use of `(..)`, parantheses, enclose the applicable SQL statement context. For example, `(SELECT)` means that the clause applies to a `SELECT` statement.
- Use of `{..}`, curly brackets, indicates an optional constant.
- Use of `<..>` indicates an optional variable.

SQL language restrictions are arranged in following groups.

- Internally Processed SQL Statements
- Removed SQL Clauses that Retain Semantic Effect
- Ignored SQL Clauses

- Translated SQL Clauses

Related Topics

- Internally Processed SQL Statements
- Removed SQL Clauses that Retain Semantic Effect
- Ignored SQL Clauses
- Translated SQL Clauses

Internally Processed SQL Statements

The following clauses are processed internally.

```
GRANT {ALL, BIND, COPY, EXECUTE, DROP, SET} ON PACKAGE <collid>.<pkgnam> TO <authid>

GRANT {ALL, BIND, COPY, EXECUTE, DROP, SET} ON PACKAGE <collid>.<pkgnam> TO <authid>
WITH GRANT OPTION

GRANT {ALL, BIND, COPY, EXECUTE, DROP, SET} ON PROGRAM <collid>.<pkgnam> TO <authid>

GRANT {ALL, BIND, COPY, EXECUTE, DROP, SET} ON PROGRAM <collid>.<pkgnam> TO <authid>
WITH GRANT OPTION

REVOKE {ALL, BIND, COPY, EXECUTE, DROP, SET} ON PACKAGE <collid>.<pkgnam> FROM
<authid>

REVOKE {ALL, BIND, COPY, EXECUTE, DROP, SET} ON PROGRAM <collid>.<pkgnam> FROM
<authid>

DROP PACKAGE <collid>.<pkgnam> VERSION <vsn>

DROP PACKAGE <collid>.<pkgnam>

FREE LOCATOR :H

HOLD LOCATOR :H
```

Removed SQL Clauses that Retain Semantic Effect

The following SQL clauses are removed from SQL statements. This should be noted because they have a semantic effect.

```
(SELECT) FOR READ ONLY

(SELECT) FOR FETCH ONLY

(DECLARE) WITH HOLD

(DECLARE) WITHOUT HOLD

(DECLARE) WITH ROWSET POSITIONING

(DECLARE) WITHOUT ROWSET POSITIONING

(DECLARE) NO SCROLL

(DECLARE) SCROLL
```

```
(CALL) WITH RETURN CLIENT

(CALL) WITH RETURN CALLER

(CALL) <:hostvar> USING DESCRIPTOR <:hostvar>

(SAVEPOINT) {UNIQUE} ON ROLLBACK RETAIN CURSORS

(SAVEPOINT) ON ROLLBACK RETAIN LOCKS

(INSERT) FOR <literal>|<bind-variable> ROWS

(INSERT) FOR MULTIPLE ROWS

(INSERT) NOT ATOMIC CONTINUE ON SQLEXCEPTION
```

Ignored SQL Clauses

These clauses are removed and ignored during translation.

```
WITH RR

WITH RR USE AND KEEP {EXCLUSIVE|UPDATE|SHARE} LOCKS

WITH RS

WITH RS USE AND KEEP {EXCLUSIVE|UPDATE|SHARE} LOCKS

WITH CS

WITH CS KEEP LOCKS

WITH UR

WITH NC
```

Translated SQL Clauses

The following SQL clauses are translated into an alternative syntax; this may have a semantic effect.

- The original clause `WHERE CURRENT OF <cursorname>` becomes `WHERE ROWID = :N`. `N` is a number.
- The original `=` becomes `'' IS NULL`.
- The original `!=` becomes `'' IS NOT NULL`.

Support for Special Registers

DB2 uses a concept known as ***special registers***; they may be thought of as environment variables within a SQL context. Oracle Database provides limited native support for special registers.

Retrieving Values from Special Registers

Example 14-2 Retrieving values from special registers

When a user enters the following statement on the client:

```
SELECT CURRENT CLIENT_ACCTNG FROM SYSIBM.SYSDUMMY1;
```

Oracle Database Provider for DRDA parses the preceding statement, and replaces the clause `CURRENT CLIENT_ACCTNG` by the clause `SYS_CONTEXT('DRDAAS_CTX_NAME','CLIENT_ACCTNG')`, as follows:

```
SELECT SYS_CONTEXT('DRDAAS_CTX_NAME','CLIENT_ACCTNG') FROM SYSIBM.SYSDUMMY1;
```

Therefore, if a translator expects a `CURRENT CLIENT_ACCTNG` clause, it will receive an altered query.

Setting Special Registers

Example 14-3 Setting special registers

When a user enters the following statement on the client:

```
SET CLIENT_ACCTNG = 'abc';
```

Oracle Database Provider for DRDA sets the value of the `CLIENT_ACCTNG` register to the string `abc` by replacing the clause `CLIENT_ACCTNG = 'abc'` by clause `SYS_CONTEXT('DRDAAS_CTX_NAME','CLIENT_ACCTNG')`, as follows:

```
SET SYS_CONTEXT('DRDAAS_CTX_NAME','CLIENT_ACCTNG') = 'abc';
```

Therefore, if a translator expects a `CURRENT CLIENT_ACCTNG` clause, it will receive an altered statement.

Special Registers Supported by Oracle Database Provider for DRDA

Oracle Database Provider for DRDA supports the following registers.

APPLICATION ENCODING SCHEME

```
CURRENT APPLICATION ENCODING SCHEME
```

Query

```
SYS_CONTEXT('DRDAAS','APPLICATION_ENCODING_SCHEME')
```

Set

Updates `SYS_CONTEXT`

Semantics

No effect

CLIENT_ACCTNG

`CURRENT CLIENT_ACCTNG CLIENT ACCTNG`

Query

`SYS_CONTEXT('DRDAAS','CLIENT_ACCTNG')`

Set

Updates `SYS_CONTEXT` and `CLIENT_INFO`

Semantics

Updates `CLIENT_INFO` in `V$SESSION`

Notes

See MVS and DDF Accounting Information, as defined by IBM and DB2, documented in the `DSNDQMDA` macro.

CLIENT_APPLNAME

`CURRENT CLIENT_APPLNAME CLIENT APPLNAME`

Query

`SYS_CONTEXT('DRDAAS','CLIENT_APPLNAME')`

Set

Updates `SYS_CONTEXT` and `CLIENT_IDENTIFIER`

Semantics

Updates `CLIENT_IDENTIFIER` in `V$SESSION`

CLIENT_PROGRAMID

`CURRENT CLIENT_PROGRAMID`

Query

`SYS_CONTEXT('DRDAAS','CLIENT_PROGRAMID')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

CLIENT_USERID

`CURRENT CLIENT_USERID CLIENT USERID`

Query

```
SYS_CONTEXT('DRDAAS','CLIENT_USERID')
```

Set

Cannot be set

Semantics

Cannot be set

CLIENT_WRKSTNNAME

```
CURRENT CLIENT_WRKSTNNAME CLIENT WRKSTNNAME
```

Query

```
SYS_CONTEXT('DRDAAS','CLIENT_WRKSTNNAME')
```

Set

Updates `SYS_CONTEXT`

Semantics

No effect

DATE

```
CURRENT DATE CURRENT_DATE
```

Query

```
CURRENT DATE
```

Set

Cannot be set

Semantics

Cannot be set

DBPARTITIONNUM

```
CURRENT DBPARTITIONNUM
```

Query

```
SYS_CONTEXT('DRDAAS','DBPARTITIONNUM')
```

Set

Cannot be set

Semantics

Cannot be set

DEBUG MODE

```
CURRENT DEBUG MODE
```

Query

```
SYS_CONTEXT('DRDAAS','DEBUG_MODE')
```

Set

Updates SYS_CONTEXT

Semantics

No effect

DECFLOAT ROUNDING MODE

```
CURRENT DECFLOAT ROUNDING MODE
```

Query

```
SYS_CONTEXT('DRDAAS','DECFLOAT_ROUNDING_MODE')
```

Set

Updates SYS_CONTEXT

Semantics

No effect

DEFAULT TRANSFORM GROUP

```
CURRENT DEFAULT TRANSFORM GROUP
```

Query

```
SYS_CONTEXT('DRDAAS','DEFAULT_TRANSFORM_GROUP')
```

Set

Updates SYS_CONTEXT

Semantics

No effect

DEGREE

```
CURRENT     DEGREE
```

Query

`SYS_CONTEXT('DRDAAS','DEGREE')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

EXPLAIN MODE

`CURRENT EXPLAIN MODE`

Query

`SYS_CONTEXT('DRDAAS','EXPLAIN_MODE')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

EXPLAIN SNAPSHOT

`CURRENT EXPLAIN SNAPSHOT`

Query

`SYS_CONTEXT('DRDAAS','EXPLAIN_SNAPSHOT')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

FEDERATED ASYNCHRONY

`CURRENT FEDERATED ASYNCHRONY`

Query

`SYS_CONTEXT('DRDAAS','FEDERATED_ASYNCHRONY')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

IMPLICIT XMLPARSE OPTION

```
CURRENT IMPLICIT XMLPARSE OPTION
```

Query

```
SYS_CONTEXT('DRDAAS','IMPLICIT_XMLPARSE_OPTION')
```

Set

Updates `SYS_CONTEXT`

Semantics

No effect

ISOLATION

```
CURRENT ISOLATION
```

Query

```
SYS_CONTEXT('DRDAAS','ISOLATION')
```

Set

Updates `SYS_CONTEXT`

Semantics

No effect

LOCK TIMEOUT

```
CURRENT LOCK TIMEOUT
```

Query

```
SYS_CONTEXT('DRDAAS','LOCK_TIMEOUT')
```

Set

Updates `SYS_CONTEXT`

Semantics

No effect

LOCALE LC_TYPE

```
CURRENT LOCALE LC_TYPE CURRENT_LC_TYPE
```

Query

`SYS_CONTEXT('DRDAAS','LC_TYPE')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

MAINTAINED TABLE TYPES FOR OPTIMIZATION

`CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION`

Query

`SYS_CONTEXT('DRDAAS','MAINTAINED_TYPES')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

MEMBER

`CURRENT MEMBER`

Query

`SYS_CONTEXT('DRDAAS','MEMBER')`

Set

Cannot be set

Semantics

Cannot be set

OPTIMIZATION HINT

`CURRENT OPTIMIZATION HINT`

Query

`SYS_CONTEXT('DRDAAS','OPTIMIZATION_HINT')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

PACKAGE PATH

`CURRENT PACKAGE PATH`

Query

`SYS_CONTEXT('DRDAAS','PACKAGE_PATH')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

PACKAGESET

`CURRENT PACKAGESET`

Query

`SYS_CONTEXT('DRDAAS','PACKAGESET')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

PATH

`CURRENT PATH CURRENT_PATH CURRENT FUNCTION PATH`

Query

`SYS_CONTEXT('DRDAAS','PATH')`

Set

Updates `SYS_CONTEXT`

Semantics

No effect

PRECISION

`CURRENT PRECISION`

Query

```
SYS_CONTEXT('DRDAAS','PRECISION')
```

Set

Updates SYS_CONTEXT

Semantics

No effect

QUERY ACCELERATION

```
CURRENT QUERY ACCELERATION
```

Query

```
SYS_CONTEXT('DRDAAS','QUERY_ACCELERATION')
```

Set

Updates SYS_CONTEXT

Semantics

No effect

QUERY OPTIMIZATION

```
CURRENT QUERY OPTIMIZATION
```

Query

```
SYS_CONTEXT('DRDAAS','QUERY_OPTIMIZATION')
```

Set

Updates SYS_CONTEXT

Semantics

No effect

REFRESH AGE

```
CURRENT REFRESH AGE
```

Query

```
SYS_CONTEXT('DRDAAS','REFRESH_AGE')
```

Set

Updates SYS_CONTEXT

Semantics

No effect

ROUTINE VERSION

```
CURRENT ROUTINE VERSION
```

Query

```
SYS_CONTEXT('DRDAAS','ROUTINE_VERSION')
```

Set

Updates `SYS_CONTEXT`

Semantics

No effect

RULES

```
CURRENT RULES
```

Query

```
SYS_CONTEXT('DRDAAS','RULES')
```

Set

Updates `SYS_CONTEXT`

Semantics

No effect

SCHEMA

```
CURRENT SCHEMA CURRENT_SCHEMA
```

Query

```
SYS_CONTEXT('USERENV','CURRENT_SCHEMA')
```

Set

Updates `SYS_CONTEXT`

Semantics

No effect

SERVER

```
CURRENT SERVER CURRENT_SERVER
```

Query

```
SYS_CONTEXT('DRDAAS','SERVER')
```

Set

Cannot be set

Semantics

Cannot be set

SQL_CCFLAGS

```
CURRENT SQL_CCFLAGS
```

Query

```
SYS_CONTEXT('DRDAAS','SQL_CCFLAGS')
```

Set

Updates `SYS_CONTEXT`

Semantics

No effect

SQLID

```
CURRENT SQLID USER
```

Query

```
USER
```

Set

Updates `SYS_CONTEXT('DRDAAS', 'CURRENT_SQLID')`

Semantics

No effect

TIMESTAMP

```
CURRENT TIMESTAMP CURRENT_TIMESTAMP
```

Query

```
CURRENT TIMESTAMP
```

Set

Cannot be set

Semantics

Cannot be set

USER

```
CURRENT USER CURRENT_USER
```

Query

```
USER
```

Set

Cannot be set

Semantics

Cannot be set

SESSION_USER

```
SESSION_USER
```

Query

```
USER
```

Set

Cannot be set

Semantics

Cannot be set

SYSTEM_USER

```
SYSTE_USER
```

Query

```
USER
```

Set

Cannot be set

Semantics

Cannot be set

ENCRYPTION PASSWORD

```
ENCRYPTION PASSWORD
```

Query

Cannot be queried

Set

Updates `SYS_CONTEXT('DRDAAS','ENCRYPTION_PASSWORD')`

Semantics

No effect

A

Scripts for Creating and Maintaining Oracle Database Provider for DRDA

Oracle Database Provider for DRDA needs several scripts to establish a proper environment.

catdrdaas.sql

The script `catdrdaas.sql` creates Oracle Database Provider for DRDA catalog objects.

```
Rem catdrdaas.sql
Rem
Rem Copyright (c) 2011, Oracle and/or its affiliates. All rights reserved.
Rem
Rem    NAME
Rem      catdrdaas.sql - CATalog Oracle Database Provider for DRDA
Rem
Rem ===========================================================================
Rem Exit immediately if there are errors in the initial checks
Rem ===========================================================================

WHENEVER SQLERROR EXIT;

DOC
#############################################################################
  Customer should create the SYSIBM tablespace

Eg:
  create tablespace SYSIBM datafile 'sysibm01.dbf'
   size 70M reuse
   extent management local
   segment space management auto
   online;

#############################################################################
#

@@prvtdpsadrda.plb
```

catnodrdaas.sql

The script `catnodrdaas.sql` removes Oracle Database Provider for DRDA catalog objects

```
Rem catnodrdaas.sql
Rem
Rem Copyright (c) 2011, 2013, Oracle and/or its affiliates.
Rem All rights reserved.
Rem
Rem    NAME
```

```
Rem      catnodrdaas.sql - CATalog NO Oracle Database Provider for DRDA
Rem

drop public synonym DBMS_DRDAAS;
drop public synonym DBMS_DRDAAS_ADMIN;

drop public synonym USER_DRDAASTRACE;
drop public synonym DBA_DRDAASTRACE;

drop public synonym ALL_DRDAASPACKAGE;
drop public synonym USER_DRDAASPACKAGE;
drop public synonym DBA_DRDAASPACKAGE;

drop public synonym USER_DRDAASPACKSTMT;
drop public synonym DBA_DRDAASPACKSTMT;

drop public synonym ALL_DRDAASPACKAUTH;
drop public synonym USER_DRDAASPACKAUTH;
drop public synonym DBA_DRDAASPACKAUTH;

drop public synonym ALL_DRDAASPACKSIDE;
drop public synonym USER_DRDAASPACKSIDE;
drop public synonym DBA_DRDAASPACKSIDE;


drop role DRDAAS_USER_ROLE;
drop role DRDAAS_ADMIN_ROLE;

drop user SYSIBM cascade;

commit;
DOC
#########################################################################
  Customer should drop the SYSIBM tablespace.

Eg:
  drop tablespace SYSIBM;

#########################################################################
```

drdapkg_db2.sql

```
Rem drdapkg_db2.sql
Rem
Rem Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
Rem
Rem    NAME
Rem      drdapkg_db2.sql  - Initialize DRDA-AS environment so that
Rem                         packages can be bound and correct datatypes
Rem                         are returned for various SQL constructs
Rem
Rem    DESCRIPTION
Rem      The DRDA-AS environment needs to be initialized so that the initial
Rem      packages (usually with an RDBCOLID of NULLID) can be bound.
Rem      Using the DataDirect driver, those package names look like
Rem          NULLID.DDOC510A, NULLID.DDOC510B, and NULLID.DDOC510C
Rem      Using the IBM driver (libdb2.so), the package names look like
Rem          NULLID.SYSTAT and NULLID.SYSshyxx (where s is one of L or S,
Rem                    h is one of H or N, y is 1, 2, 3, or 4 and
```

```
Rem                                          xx is somwhere in 00 through FF)
Rem      Also, various columns must be TYPEMAPped -- their normal
Rem      attributes must be altered.
Rem
Rem      The initial package bindings should be done under the id that runs
Rem      this script. That is, if we run this script under the Oracle ID
Rem      of xxxx, then the initial connection through an ODBC driver should be
Rem      using that same id, namely xxxx.
Rem
Rem    NOTES
Rem      The following is relevant when using the IBM driver: libdb2.so ...
Rem        Note that the normal set of packages produced by the jdbcbinder
Rem        process (db2jdbcbinder in DB2/LUW) defines packages with names like
Rem        SYSSTAT and SYSLNmnn and SYSLHmnn. Thus, before running the
Rem        jdbcbinder on DB2/LUW specifying the Oracle Id accepted in the prompt
Rem        for this script, one needs to inform DRDA-AS that the id has the
Rem        required privilege, namely to create ANY package in the NULLID
Rem        schema. That is part of what we are doing here.
Rem
Rem      This script can be run ONLY by a user that has the ability to use
Rem      the DRDAAS_ADMIN_ROLE which must be GRANTed to the user; also this
Rem      role must be active either by being set as a default ROLE or
Rem      by actively doing a SET ROLE DRDAAS_ADMIN_ROLE.

SET ECHO ON
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100

SET SERVEROUTPUT ON

SHOW USER

/* The following  will work even if DRDAAS_ADMIN_ROLE is not one of the */
/* DEFAULT Roles, but HAS been GRANTed to the user running this script. */
/* (A prerequisite of setting a DEFAULT ROLE for a user is that the user*/
/*  has been GRANTed that ROLE).                                        */

SET ROLE DRDAAS_ADMIN_ROLE;

Prompt Enter the OracleID under which the initial package BINDs will be made
Prompt Use quotes (') if needed.
Accept OracleID
Define BindID = &OracleID
Prompt Enter default collection ID for package binding (usually NULLID)
Prompt Use quotes (') if needed.
Accept DefaultCollection
Define DfltCollid = &DefaultCollection

declare
 id_passed CONSTANT VARCHAR2(128) := '&&BindId';
 collid_passed CONSTANT VARCHAR2(128) := '&&DfltCollid';
 id_to_use VARCHAR2(128);
 collid_to_use VARCHAR2(128);
 id_len PLS_INTEGER;
 collid_len PLS_INTEGER;
 quote CONSTANT CHAR := '''';
begin
```

```
id_len := LENGTH(id_passed);
collid_len := LENGTH(collid_passed);

IF SUBSTR(id_passed, 1, 1) = quote AND SUBSTR(id_passed, id_len, 1) = quote
     THEN
   /* Use Id exactly as passed */
   id_to_use := SUBSTR(id_passed, 2, id_len - 2);
ELSE
   id_to_use := UPPER(id_passed) ;
END IF;
IF SUBSTR(collid_passed, 1, 1) = quote AND
   SUBSTR(collid_passed, collid_len, 1) = quote THEN
   /* Use Collection ID  exactly as passed */
   collid_to_use :=  SUBSTR(collid_passed, 2, collid_len-2) ;
ELSE
   collid_to_use :=  UPPER(collid_passed) ;
END IF;

-- The following section is pertinent to ALL flavors of DB2
-- =======================================================

 -- The id of the specified  user will have ALL privileges for ANY Package in
 -- RDBCOLID=collid_to_use

 DBMS_DRDAAS_ADMIN.GRANT_PRIVILEGE( DBMS_DRDAAS_ADMIN.ALL_PRIVILEGE,
                                    collid_to_use,'*', id_to_use);

 commit;

 -- If you might want the id specified to create packages in rcbcolid= SCOTT,
 -- then you need to do the following:

 -- DBMS_DRDAAS_ADMIN.GRANT_PRIVILEGE(DBMS_DRDAAS_ADMIN.ALL_PRIVILEGE,
 --                                     'SCOTT','*', id_to_use);
 -- commit;

 -- Typemaps ...

 -- The described "type" for "COUNT(*)" columns in any package in the
 -- collid_to_use schema should be INTEGER

 -- General "COUNT(*)" case
 DBMS_DRDAAS_ADMIN.SET_TYPEMAP(collid_to_use,'*','COUNT(*)',
                               'NUMBER=INTEGER');



 -- ================================================
 -- The following section is pertinent ONLY to DB2/zOS
 -- ================================================

 -- To use DB2 z/OS SPUFI asgainst DRDAAS, the given oracle-id must be able to
 -- define packages in the DSNESPCS and DSNESPRR schemas
 --
 -- DB2 z/OS SPUFI Packages

 DBMS_DRDAAS_ADMIN.GRANT_PRIVILEGE( DBMS_DRDAAS_ADMIN.ALL_PRIVILEGE,
                                   'DSNESPCS','*', id_to_use);
 DBMS_DRDAAS_ADMIN.GRANT_PRIVILEGE( DBMS_DRDAAS_ADMIN.ALL_PRIVILEGE,
                                   'DSNESPRR','*', id_to_use);
 commit;
```

```
-- ========================================================================
-- Specific DataDirect ODBC package discovery queries for DB2 z/OS

-- When accessing the collid_to_use.DDOC510A package, the
-- "column" MAX(SECTNO) referencing table SYSIBM.SYSPACKSTMT (which is a
--  NUMBER in Oracle terms) should be described as a SMALLINT to the
-- application

DBMS_DRDAAS_ADMIN.SET_TYPEMAP(collid_to_use,'DDOC510A',
                              'SYSIBM.SYSPACKSTMT:MAX(SECTNO)',
                              'NUMBER=SMALL');

-- Same as above but for package collid_to_use.DDOC510B

DBMS_DRDAAS_ADMIN.SET_TYPEMAP(collid_to_use,'DDOC510B',
                              'SYSIBM.SYSPACKSTMT:MAX(SECTNO)',
                              'NUMBER=SMALL');


-- When accessing the collid_to_use.DDOC510A  package, the
-- "column" COUNT(DISTINCT(NAME)) referencing table SYSIBM.SYSPACKSTMT
-- (which is  a NUMBER in Oracle terms) should be described as a SMALLINT to
--  the  application

DBMS_DRDAAS_ADMIN.SET_TYPEMAP(collid_to_use,'DDOC510A',
                              'SYSIBM.SYSPACKSTMT:COUNT(DISTINCT(NAME))',
                              'NUMBER=INTEGER');

-- Same as above but for package collid_to_use.DDOC510B

DBMS_DRDAAS_ADMIN.SET_TYPEMAP(collid_to_use,'DDOC510B',
                              'SYSIBM.SYSPACKSTMT:COUNT(DISTINCT(NAME))',
                              'NUMBER=INTEGER');
commit;

-- ========================================================================
-- Specific DataDirect JDBC package discovery queries for DB2 z/OS

-- When accessing the collid_to_use.DDJC360B package, the "column"
-- COUNT(*)-1 referencing table SYSIBM.SYSPACKSTMT (which is  a NUMBER in
-- Oracle terms) should be described as an INTEGER to the application

DBMS_DRDAAS_ADMIN.SET_TYPEMAP(collid_to_use,'DDJC360B',
                             'SYSIBM.SYSPACKSTMT:COUNT(*)-1',
                             'NUMBER=INTEGER');

-- When accessing the collid_to_use.DDJC360B package, the "column"
-- COUNT(*)-1 referencing table SYSIBM.SYSPACKSTMT (which  might be described
-- as NUMBER(0,-127) in Oracle terms) describe the column as an INTEGER
-- to the application.

DBMS_DRDAAS_ADMIN.SET_TYPEMAP(collid_to_use,'DDJC360B',
                              'SYSIBM.SYSPACKSTMT:COUNT(*)-1',
                              'NUMBER(0,-127)=INTEGER');
commit;


-- ================================================
--  the following section is pertinent ONLY to DB2/luw
-- ================================================
```

```
  -- ========================================================================
  -- Specific DataDirect ODBC package discovery queries for DB2/LUW

  DBMS_DRDAAS_ADMIN.SET_TYPEMAP(collid_to_use,'DDOC510A',
                                'SYSIBM.SYSPLAN:MIN(TOTALSECT)',
                                'NUMBER=SMALL');

  DBMS_DRDAAS_ADMIN.SET_TYPEMAP(collid_to_use,'DDOC510A',
                                'SYSIBM.SYSPLAN:COUNT(*)',
                                'NUMBER=INTEGER');
  --  DBMS_DRDAAS_ADMIN.SET_TYPEMAP(collid_to_use,'DDOC510A',
  --                                'MIN(TOTALSECT)', 'NUMBER=SMALL');
  commit;

  --  ================================================
  --  The following section is pertinent ONLY to DB2/iOS
  --  ================================================

  --  Currently empty!
end;
/
```

drdasqtt_translator_setup.sql

```
Rem drdasqtt_translator_setup.sql
Rem
Rem Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
Rem
Rem    NAME
Rem      drdasqtt_translator_setup.sql - Generalized script for setting up an
Rem                                      external SQL translator
Rem
Rem
Rem    DESCRIPTION
Rem      This script can be used to set up any external SQL translator.
Rem      Some translators, e.g., BableFish, may need extra customizations.
Rem      For BabelFish, that would include the source/target SQL text for
Rem      the fingerprint translations (to be inserted into
Rem      DBA_SQL_TRANSLATIONS).
Rem
Rem    NOTES
Rem      Should be run "/ as sysdba"
Rem

SET ECHO ON
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET SERVEROUTPUT ON

show user

Prompt Enter schema in which the SQL Translator Interface Package will be created as
well
Prompt as into which the third-party SQL translator will be loaded (usually SYSIBM).
```

```
Accept  TRANS_PKG_SCHEMA_ Prompt 'SQL Translator Interface Package Schema:'
DEFINE TRANSLATOR_PACKAGE_SCHEMA =  &TRANS_PKG_SCHEMA_

Prompt Enter unqualified name of the SQL Translator Interface Package
Accept  TRANS_PKG_NAME_ Prompt 'SQL Translator Interface Package Name:'
DEFINE TRANSLATOR_PACKAGE_NAME = &TRANS_PKG_NAME_

Prompt Enter schema in which the Translation Profile will be created:
Accept  TRANS_PROFILE_SCHEMA_ Prompt 'Translation Profile Schema:'
DEFINE TRANS_PROFILE_SCHEMA = &TRANS_PROFILE_SCHEMA_

Prompt Enter the unqualified name of the translation profile:
Accept  TRANS_PROFILE_NAME_ Prompt 'Translation Profile Name:'
DEFINE TRANS_PROFILE_NAME =  &TRANS_PROFILE_NAME_

Prompt Enter the "language" type of the translator: C, java, etc
Accept TRANS_LANG_ Prompt 'Translator Language:'
DEFINE TRANS_LANG = &TRANS_LANG_

Prompt Enter the names of the third-party SQL Translator objects;
Prompt They should be available through rdbms/drdaas/jlib/..
Prompt If there is more than one object, enclose the entire set
Prompt in four quotes, such as  ''''object_a object_b''''.
Accept  EXTERNAL_CODE_ Prompt 'SQL Translator object(s):'
DEFINE EXTERNAL_CODE  = '''&EXTERNAL_CODE_'''
DEFINE EXTERNAL_CODE


Prompt Enter the signature for the entry for 'translateSQL' in one of the
Prompt  previously entered SQL Translator objects:
Accept CALLOUT_TRANSLATE_SQL_ Prompt 'Entry for translateSQL:'
DEFINE CALLOUT_TRANSLATE_SQL = '''&CALLOUT_TRANSLATE_SQL_'''

Prompt Enter the signature for the entry for 'translateError' in one of the
Prompt previously entered SQL Translator objects
Accept CALLOUT_TRANSLATE_ERROR_ Prompt 'Callout for translateError:'
DEFINE CALLOUT_TRANSLATE_ERROR = '''&CALLOUT_TRANSLATE_ERROR_'''


Rem Create the SQL Translator Interface Package ...

create or replace package &&TRANSLATOR_PACKAGE_SCHEMA..&&TRANSLATOR_PACKAGE_NAME as

  procedure translate_sql(sql_text             in  clob,
                          translated_text      out clob);

  procedure translate_error(error_code         in  binary_integer,
                            translated_code    out binary_integer,
                            translated_sqlstate out varchar2);

end;
/
show errors

declare
  COMP_ERROR exception;
  pragma EXCEPTION_INIT(COMP_ERROR, -24344);
  translateSQLcode CONSTANT VARCHAR2(1024) := &&CALLOUT_TRANSLATE_SQL;
  translateErrorcode CONSTANT VARCHAR2(1024) := &&CALLOUT_TRANSLATE_ERROR;
  translateSQLToUse VARCHAR2(1024);
  translateErrorToUse VARCHAR2(1024);
```

```
  ln NUMBER;
  quote CONSTANT CHAR := '''';
  my_cursor BINARY_INTEGER;
  n BINARY_INTEGER;
  i BINARY_INTEGER;
  package_body   VARCHAR2(1024);
  /* we can't use bind variables to substitute for the "name" part of the */
  /* procedures in the package body declaration; the "name" part MUST be a*/
  /* single-quoted string!!! ARGHHH !!!                                   */
  package_body_1 VARCHAR2(400) :=
       'create or replace package ' ||
          'body &&TRANSLATOR_PACKAGE_SCHEMA..&&TRANSLATOR_PACKAGE_NAME as '  ||
            'procedure translate_sql(sql_text             in  clob, '       ||
                                    '  translated_text      out clob) as '    ||
              'language &&TRANS_LANG '                                       ||
              'name ''';
  package_body_2 VARCHAR2(400) := ''' ;'                                    ||
            'procedure translate_error(error_code  in  binary_integer, '     ||
                                      ' translated_code out binary_integer,' ||
                                     ' translated_sqlstate out varchar2) as ' ||
              'language &&TRANS_LANG '                                       ||
              'name ''';
  package_body_end VARCHAR2(10) := '''; end;';
begin
  ln := LENGTH(translateSQLcode);
  IF SUBSTR(translateSQLCode,1,1) = quote AND
     SUBSTR(translateSQLCode,ln,1) = quote THEN
    translateSQLToUse := SUBSTR(translateSQLCode, 2, ln-2);
  ELSE
    translateSQLToUse := translateSQLCode;
  END IF;
  ln := LENGTH(translateErrorcode);
  IF SUBSTR(translateErrorCode,1,1) = quote AND
     SUBSTR(translateErrorCode,ln,1) = quote THEN
    translateErrorToUse := SUBSTR(translateErrorCode, 2, ln-2);
  ELSE
    translateErrorToUse := translateErrorCode;
  END IF;
  my_cursor := DBMS_SQL.OPEN_CURSOR;
  package_body := package_body_1 || translateSQLToUse || package_body_2 ||
                  translateErrorToUse || package_body_end;
  BEGIN
    DBMS_SQL.PARSE(my_cursor, package_body, DBMS_SQL.NATIVE);
  EXCEPTION
    when COMP_ERROR THEN DBMS_OUTPUT.PUT_LINE('SQLCODE=' || SQLCODE || ':' ||
                                              SQLERRM);
  END;
  n := DBMS_SQL.EXECUTE(my_cursor);
  DBMS_SQL.CLOSE_CURSOR(my_cursor);
end;
/

show errors

Rem Load the Java code
Rem CALL DBMS_JAVA.LOADJAVA('-definer -genmissing -schema SYSIBM
Rem rdbms/drdaas/jlib/DBTooSQLAPI.jar rdbms/drdaas/jlib/DBTooTranslator.class',
Rem '((* SYSIBM)(* PUBLIC)(* -))');
Rem DBTooSQLAPI.jar and DBTooSQLTranslator.class are fictional names

set serveroutput on
```

```
show user

Rem Load the .class and .jar objects as specified ...

declare
  extcode VARCHAR2(4096) := &&EXTERNAL_CODE;
  real_extcode VARCHAR2(4096);
  first_parm_first_part VARCHAR2(128) :=
                  '-definer -genmissing -schema &&TRANSLATOR_PACKAGE_SCHEMA ';
  first_parm VARCHAR2(4096);
  ln NUMBER;
begin
  ln := LENGTH(extcode);
  /* We might have a beginning and ending aprostrophe --*/
  /* we need to delete them                           */
  IF SUBSTR(extcode,1,1) = '''' AND SUBSTR(extcode,ln,1) = '''' THEN
     real_extcode := SUBSTR(extcode, 2, ln-2);
  ELSE
     real_extcode := extcode;
  END IF;
  first_parm := first_parm_first_part || ' ' || real_extcode;
/*DBMS_OUTPUT.PUT_LINE('First parm ' || first_parm); */
  DBMS_JAVA.LOADJAVA(first_parm ,
                     '((* &&TRANSLATOR_PACKAGE_SCHEMA)(* PUBLIC)(* -))');
end;
/

GRANT EXECUTE ON &&TRANSLATOR_PACKAGE_SCHEMA..&&TRANSLATOR_PACKAGE_NAME  to PUBLIC;

GRANT CREATE SQL TRANSLATION PROFILE TO &&TRANS_PROFILE_SCHEMA;
GRANT TRANSLATE ANY SQL TO &&TRANS_PROFILE_SCHEMA WITH ADMIN OPTION;

CALL
DBMS_SQL_TRANSLATOR.DROP_PROFILE('&&TRANS_PROFILE_SCHEMA..&&TRANS_PROFILE_NAME');
CALL
DBMS_SQL_TRANSLATOR.CREATE_PROFILE('&&TRANS_PROFILE_SCHEMA..&&TRANS_PROFILE_NAME');


begin

dbms_sql_translator.set_attribute('&&TRANS_PROFILE_SCHEMA..&&TRANS_PROFILE_NAME',
                        dbms_sql_translator.attr_translator,

'&&TRANSLATOR_PACKAGE_SCHEMA..&&TRANSLATOR_PACKAGE_NAME');

 dbms_sql_translator.set_attribute('&&TRANS_PROFILE_SCHEMA..&&TRANS_PROFILE_NAME',
                        dbms_sql_translator.attr_translate_new_sql,
                                       dbms_sql_translator.attr_value_true);
end;
/

GRANT ALL ON SQL TRANSLATION PROFILE &&TRANS_PROFILE_SCHEMA..
   &&TRANS_PROFILE_NAME TO &&TRANSLATOR_PACKAGE_SCHEMA ;
GRANT USE ON SQL TRANSLATION PROFILE &&TRANS_PROFILE_SCHEMA..
   &&TRANS_PROFILE_NAME TO DRDAAS_USER_ROLE;
```

drdasqt_set_profile_dd.sql

```
Rem drdasqt_set_profile_dd.sql
Rem
Rem Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
Rem
Rem    NAME
Rem      drdasqt_set_profile_dd.sql - Set a sqllangprofile for each of
Rem                                   the DataDirect (dd) packages.
Rem

SET ECHO ON
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 80
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100

Rem You will be prompted for the profile name.
Rem Must be run under an id that has access to the DRDAAS_ADMIN_ROLE role.

Rem set echo on
set serveroutput on


SET ROLE DRDAAS_ADMIN_ROLE;

prompt Enter the (qualified) profile name to use for DataDirect packages
Accept SQLPROFILENAME
Define PROFILE_NAMEX = &SQLPROFILENAME
prompt Enter the default Package Collection (usually NULLID)
Prompt Use quotes (') if needed
Accept DefaultCollection
Define PACKAGE_COLLECTIONX = &DefaultCollection

declare
  TYPE FIRST_CHAR  IS VARRAY(4) of CHAR(1);
  TYPE SECOND_CHAR IS VARRAY(3) of CHAR(1);

  first_chr  FIRST_CHAR := FIRST_CHAR();
  second_chr SECOND_CHAR := SECOND_CHAR();
  package_name VARCHAR2(128);
  profile_name CONSTANT VARCHAR2(128) := '&&PROFILE_NAMEX';
  package_collection_as_passed CONSTANT VARCHAR2(128) :=
                                        '&&PACKAGE_COLLECTIONX';
  package_collection VARCHAR2(128);
  cmd VARCHAR2(255);
  quote CONSTANT CHAR := '''';
  ln BINARY_INTEGER;
begin

  ln := LENGTH(package_collection_as_passed);
  IF SUBSTR(package_collection_as_passed, 1,  1) = quote AND
     SUBSTR(package_collection_as_passed, ln, 1) = quote THEN
    /* Use package_collection exactly as passed */
     package_collection := SUBSTR(package_collection_as_passed, 2, ln - 2);
  ELSE
     package_collection := UPPER(package_collection_as_passed) ;
  END IF;

  first_chr.EXTEND(4);
  first_chr(1) := 'C'; first_chr(2) := 'S';
```

```
  first_chr(3) := 'U'; first_chr(4) := 'R';
  second_chr.EXTEND(3);
  second_chr(1) := 'A'; second_chr(2) := 'B'; second_chr(3) := 'C';

  FOR f IN 1..first_chr.COUNT
  LOOP
    FOR s IN 1..second_chr.COUNT
    LOOP
       package_name := 'DDO' || first_chr(f) || '510' || second_chr(s);

       cmd := 'DBMS_DRDAAS_ADMIN.SET_PROFILE(' || package_collection || ',' ||
                                                  package_name  || ',' ||
                                                  profile_name  || ')';
       DBMS_OUTPUT.PUT_LINE('Doing ' || cmd);
       DBMS_DRDAAS_ADMIN.SET_PROFILE(package_collection, package_name,
                                     profile_name);

    END LOOP;
  END LOOP;
end;
/
```

B

Package Binding Options in Oracle Database Provider for DRDA

DB2 DSN sub-command `BIND PACKAGE` has several options that are used when Oracle binds a client application' package.

Further details on the `BIND PACKAGE` sub-command are part of the following volume: *DB2® 10 for z/OS® Command Reference (SC19-2972-05)*.

CONCURRENTACCESSRESOLUTION

Always implements `WAITFOROUTCOME` semantics; this is fundamental to how Oracle does data locking, and cannot be changed.

CURRENTDATA

Supports `YES` semantics.

DBPROTOCOL

Supports `DRDA` semantics.

DEFER/NODEFER

Supports `NODEFER` semantics.

DYNAMICRULES

Supports `RUN` semantics.

Default qualifier usage applies for each statement, as defined by DB2's rules. This is equivalent to `SET CURRENT SQLID` for a `STATIC` or `DYNAMIC SQL` statement. However, Oracle restricts object access authorization to the current logon id, and not the package owner's id. To control object access, the logon must be handled either through an explicit access grant (object grants, or role grants), or through a stored procedure that imposes object access authorization.

ENABLE/DISABLE

Supports `ENABLE` semantics.

EXTENDEDINDICATOR

Supports `NO` semantics.

ISOLATION

Supports `CS` semantics.

Oracle, and therefore Oracle Database Provider for DRDA, do not support most isolation modes. Oracles data isolation mode may be described as a compromise between CS and RR. This cannot be changed because it is fundamental to how

Oracle implements data integrity management. While the value is ignored, it is stored with the package definition for future processing.

KEEPDYNAMIC

Supports `YES` semantics.

OWNER

Authorization id must be a valid Oracle userid.

QUALIFIER

Qualifier name should be a valid schema name within Oracle.

RELEASE

Implements `COMMIT` semantics.

When using DRDA, the release of resources is performed at different level. At Commit/Rollback, the semantics of the cursor or statement may request a release of cursor and object locks. Other resources, such as the package itself, are retained until the session ends, and are only then de-allocated. This behaviors is specified largely by the client at runtime, rather than by the server.

REOPT

Implements `AUTO` semantics.

Oracle, by default, automatically evaluates execution plans based on statement and host variables; both `STATIC` or `DYNAMIC` statements receive the same treatment.

ROUNDING

Implements `HALFEVEN` semantics.

VALIDATE

Implements `RUN` semantics.

This option is ignored because Oracle Database Provider for DRDA does not perform validation of `STATIC SQL` statements.

Glossary

Application requester (AR)

Used by the application, AR assumes the client component in a classical client/server configuration acting on the behalf of the application making all DRDA protocol requests.

Application server (AS)

Application Server. Assumes the Server component in a classical client/server configuration, acting as the DRDA protocol front end for the server, processing DRDA requests, performing server function calls and returning results to the client.

AS/400

Application System/400, an IBM platform.

CCSID

Coded Character Set Identifier (IBM/DRDA terminology). A 16bit number that includes a specific set of encoding scheme identifiers, character set identifiers code page identifiers, and other information that uniquely identifies the coded graphic character representation.

Examples include: 500 INTL EBCDIC [CECP: Belgium, Canada (AS/400*), Switzerland, International Latin1], 819 ISO 88591 ASCII [ISO 88591: Latin Alphabet Number 1Latin1 countries and regions], 850 LATIN1 PCDATA [PC Data: MLP 222 Latin Alphabet Number 1Latin1 Countries and Regions]

Special cases: CCSID 65534 (defer codepage to lower level definition) and CCSID 65535 (binary data).

Database Request Module (DBRM)

(IBM/DRDA terminology). This is a proprietary on-disk file that contains the SQL statements of an embedded SQL application after it has been preprocessed. The post-processed application source will then contain only a statement reference number indicating the SQL statement to be used in the DBRM. The statements are externalized so that the database system can fully analyze and optimize execution of the SQL.

DB2

Short for IBM DB2, the database produced by IBM.

DBCS

In IBM terminology, this is a Doublebyte Character Set, any character set that has character code points of exactly 2 bytes in length.

Dedicated Instance Configuration

an Oracle Database Provider for DRDA configuration where a single `RDB_MAP` entry is made for each Oracle Database Provider for DRDA instance. This approach is used for IBM DB2 Database for z/OS DRDA clients. See Multiplexed Instance Configuration.

Distributed Data Management (DDM)

DDM architecture provides the overall command and reply structure used by the distributed database. Fewer than 20 commands are required to implement all of the distributed database functions for communication between the application requester (client) and the application server.

Distributed Relational Database Architecture

Distributed Relational Database Architecture (DRDA) is an open, published architecture that enables communication between applications and database systems on disparate platforms, whether those applications and database systems are provided by the same or different vendors and whether the platforms are the same or different hardware/software architectures. DRDA is a combination of other architectures and the environmental rules and process model for using them. The architectures that actually comprise DRDA are Distributed Data Management (DDM) and Formatted Data Object Content Architecture (FD:OCA).

DRDA

See Distributed Relational Database Architecture.

DRDA Package

(IBM/DRDA terminology). A "package" is a collection of SQL statements and attributes defined in an embedded SQL application. A Package is created by binding a resource file (DBRM) though DRDA BIND request commands.

Dynamic SQL

A Package may contain a mix of Static SQL and Dynamic SQL. Dynamic SQL is SQL that is not preformed in the application (and generally considered *adhoc*, even if it is constructed by the application). The primary differences between static and dynamic SQL is that static SQL is preloaded into the database as part of the Package and dynamic SQL must be sent to the database a runtime for execution. Historically, Oracle only implements dynamic SQL, relying on the strength of it's cursor caching facility to enhance the speed of SQL execution which largely negates the need for static SQL. See Static SQL.

FD:OCA

See Formatted Data Object Content Architecture.

Formatted Data Object Content Architecture

The Formatted Data Object Content Architecture (FD:OCA) provides the data definition architectural base for DRDA. Descriptors defined by DRDA provide layout and datatype information for all the information routinely exchanged between the application requesters and servers. A descriptor organization is defined by DRDA to allow dynamic definition of user data that flows as part of command or reply data. DRDA also specifies that the descriptors only have to flow once per answer set, regardless of the number of rows actually returned, thus minimizing data traffic on the wire.

IBM

International Business Machines, the company responsible DB2 Database and DB2 group of products.

MBCS

In IBM terminology, this is a Multibyte Character Set, any character set that may contain variable-length character code points. An example is Unicode (UTF-8). Another example is when a singlebyte character set and a doublebyte character set are combined to make a multibyte character set.

Multiplexed Instance Configuration

an Oracle Database Provider for DRDA configuration where additional `DATA_PORT` entries may be specified with different host name or IP addresses, and unallocated network port numbers. See Dedicated Instance Configuration.

Oracle Call Interface (OCI)

OCI is a set of C-language software APIs that provide an interface to the Oracle database.

OCI consists of procedural APIs that perform database administration tasks and for using PL/SQL or SQL to query, access, and manipulate data.

SBCS

In IBM terminology, this is a Singlebyte Character Set, any character set that has character code points of exactly 1 byte in length.

Special Register

In DB2, a special register is a storage area defined for an application process, and used to store information that can be referenced in SQL statements. A reference to a special register is a reference to a value provided by the current server. If the value is a string, its CCSID is a default CCSID of the current server.

Static SQL

A Package may contain a mix of Static SQL and Dynamic SQL. Static SQL is SQL that is already written as part of the application. It's syntax and use of in-program variables (bind variables) is fixed in the code and cannot be changed during execution. Also, the SQL is not part of the application. It is extracted from the source and uploaded as part of the Package, further fixing its form and preventing malicious modification of it's

function. It is presumed that static SQL is also heavily analyzed in advance to allow for a more efficient execution plan. Historically, DB2 invented static SQL for applications that were primarily batch oriented and thus not subject to the idea of dynamic construction. See Dynamic SQL.

SQL Type

In IBM's DB2 terminology, this is an interface value for datatype.

SQLAM Level

SQLAM stands for SQL Application Manager. SQLAM Level 8 is the support level provided by Oracle's implementation of DRDA.

UOW

In IBM terminology, this is a Unit of Work, a resource designation for all changes to a database within the scope of a transaction. DRDA maintains a Distributed (or Remote) Unit of Work between a client and a database for the duration of the application's transaction.

Index

O

P

Q

R

S

T

U

V

W

X

Z