Oracle Database Gateways

An Oracle White Paper
July 2007

ORACLE

Oracle Database Gateways

INTRODUCTION

There is a great demand to integrate and consolidate all information for an organization from a central point. Integration and consolidation of information in this manner allows an organization to easily and quickly take advantage of the synergies inherent in business information. The challenge is to quickly, efficiently, and economically deploy corporate data that may exist on many Oracle and non-Oracle systems through a single application, providing the customer with a comprehensive view of the corporate information. A technical solution is required that allows the IT professional to easily publish the corporate data, regardless of the non-Oracle system, operating system, or networking platform.

Oracle offers Database Gateways for synchronously sharing information within an enterprise. Synchronous access uses Oracle's Distributed SQL features to consolidate data on the fly, masking the location of data from the application or user by making it appear as a local table.

Database Gateways provide the flexibility, power, and scalability to transparently access any number of non-Oracle systems, including SQL Server, DB2, Teradata, Sybase, IMS, VSAM, Adabas and WebSphere MQ from an Oracle environment. Application developers can focus their efforts on developing business solutions instead of trying to develop custom solutions to access non-Oracle systems.

As a result of the highly transparent nature of Database Gateways, access to both local and remote data is achieved using Oracle SQL and procedure call interfaces, even if the remote data is stored in multiple vendors' databases. This transparency eliminates the need for application developers to customize their applications to access data from different non-Oracle systems, thus decreasing development efforts and increasing the mobility of the application.

CONNECTING DISPARATE SYSTEMS

Although the user interfaces for different non-Oracle systems based on SQL standards may appear to work identically, there may be subtle (and not so subtle) differences between these non-Oracle systems. These differences may prevent the disparate systems from interoperating effectively.

Generally, there are three areas of operation that may prevent effective interoperation: SQL translation, data dictionary translation, and datatype

translation. A solution is required that will be able to effectively translate these operations from one system's dialect to another.

SQL Translations

Even though a relational data store may be based on SQL standards, there may be subtle differences between manufacturers in the implementation. For example, if you wanted the following result set to be in uppercase letters, you would execute the following SQL statement in an Oracle environment:

```
SELECT TO_UPPER(ename) FROM emp;
```

In a non-Oracle environment, however, you might execute the following SQL statement to retrieve the same results:

```
SELECT UPPERCASE(ename) FROM emp;
```

The heterogeneous solution should automatically translate the dialect of the foreign system to that of the local system (where the transaction originated), eliminating the need for the user to utilize more than one system's dialect. This solution should also occur transparently.

Data Dictionary Translations

Metadata, or information about a database environment, is extremely useful to DBAs managing a database environment. Different manufacturers have their own methods of storing this data in a *data dictionary* and displaying this data. A mechanism is required that would allow a query of the metadata at a remote disparate system to be displayed in the format of the local system.

For example, a DBA might issue the following SQL statement to view all tables in an Oracle database:

```
SELECT * FROM sys.dba_objects
WHERE object_type = 'TABLE';
```

In a non-Oracle environment, the following SELECT statement might yield the same results:

```
SELECT * FROM catalog_objects
WHERE object_name LIKE '%EP%'
AND object_type = 'TABLE';
```

Just as for SQL translations, the heterogeneous solution should automatically translate a data dictionary query in the dialect of the local database to that of the target remote database. This translation involves rewriting the local SELECT statement into a query that will produce the same results by querying the remote system.

Datatype Translations

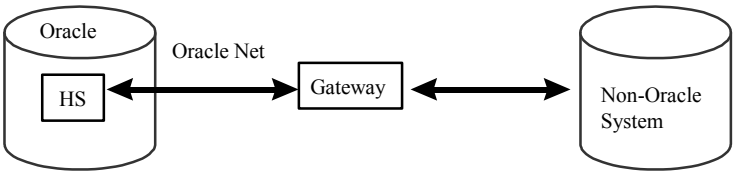
The final area that affects the interoperation of disparate systems involves translating one manufacturer's datatype to another manufacturer's. For example,

the heterogeneous solution should transparently handle translating a DB2 PACKED DECIMAL datatype to an Oracle NUMBER datatype.

When a DESCRIBE statement is issued for a particular remote object, it should be described with the datatypes of the local database.

ORACLE'S SYNCHRONOUS INFORMATION INTEGRATION SOLUTION

To meet the requirements described in the "Connecting Disparate Systems" section, Oracle provides Database Gateways. This solution is a combination of the *Heterogeneous Services* (HS) component that is integrated in the Oracle database and a gateway, which provides information for and connectivity to non-Oracle systems.



Heterogeneous Services

The processing power for communicating with a non-Oracle system is an integrated component of the Oracle database known as *Heterogeneous Service*s. Heterogeneous Services (HS) extends the Oracle SQL engine to recognize the SQL and procedural capabilities of the remote non-Oracle system and the mappings required to obtain necessary data dictionary information. As an integrated component, Heterogeneous Services can take advantage of the powerful SQL parsing and distributed optimization capabilities of the Oracle database. Additionally, the transaction coordination of the database maintains the transaction coordination between Oracle and the remote non-Oracle system, such as providing the two-phase commit protocol to ensure distributed transaction integrity, even for non-Oracle systems that do not natively support two-phase commit. This integration also ensures that the gateway can immediately take advantage of any enhancements made to future releases of the Oracle database.

As an integrated component of the Oracle database, Heterogeneous Services provides the following services:

- Transaction Service
- SQL Service
- Procedural Service

In addition to the above services, Heterogeneous Services provides the ability to issue native SQL to the target non-Oracle system. This feature is known as *Pass-Through SQ*L.

Transaction Service

The *transaction service* ensures that the Oracle environment can manage authenticated sessions with non-Oracle systems. This session management includes establishing the connection when the non-Oracle system is initially accessed and transparently closing the connection with the non-Oracle system when the Oracle client session is complete.

The transaction service also ensures that distributed transactions will maintain global data integrity. Specifically, Oracle uses a two-phase commit protocol to maintain global data integrity, even if the non-Oracle system does not natively support a two-phase commit protocol.

SQL Service

The *SQL service* provides the necessary translation capabilities to address two of the three types of translations that are needed to connect disparate systems (see the "Connecting Disparate Systems" section): SQL and data dictionary translations.

- **SQL Translation**: The SQL service provides the translation capabilities to translate Oracle SQL into the proper SQL dialect of the non-Oracle system. The SQL service uses the information provided by the gateway to translate Oracle SQL to the appropriate dialect.
- **Data Dictionary Translation**: The SQL service rewrites any reference to a data dictionary table in the query of a non-Oracle system so that the result set appears as if it were retrieved from an Oracle data dictionary.

The third type of translation, datatype translation, is handled by the gateway and is described in the "Datatype Translation" section.

Procedural Service

Procedural service provides a programmatic interface for executing stored procedures on a non-Oracle system. Procedural service, for example, enables an Oracle environment to access procedural systems, such as a messaging or queuing system. Specifically, procedural service maps a PL/SQL call into an equivalent procedure or function call on the non-Oracle system.

Pass-Through SQL

Pass-through SQL provides the facility to issue native SQL against a non-Oracle system. This flexibility enables you to execute functions or procedures on the non-Oracle system that are not supported by the gateway. Pass-through SQL supports both result sets and bind variables. *Pass-through SQL* can be used to perform DDL on the non-Oracle system.

Heterogeneous Services Benefits

As previously described, the integration of Heterogeneous Services with the Oracle database is beneficial for a heterogeneously distributed environment. Most notable of these benefits are:

- Database Integration
- Distributed Optimization

Database Integration

When a gateway is registered with the Oracle database, the capabilities and translations of the non-Oracle system are retrieved and stored locally. Locally storing this information eliminates the need for the Oracle database to fetch this information each time a connection is established with the non-Oracle system, resulting in a reduction of round-trips and data transferred necessary to establish the connection with the non-Oracle system.

Access to the non-Oracle system is enhanced, by using the highly efficient SQL parser contained in the Oracle database. Additionally, by using the SQL parser in the database, the issued SQL statement only needs to be parsed once (compared with a local parse and then again in a stand-alone gateway which is the architecture of other products in this arena).

Manageability of a heterogeneous environment has also been enhanced by the addition of V$ views to retrieve information about the heterogeneous environment.

Distributed Optimization

The distributed SQL optimization capabilities of the Oracle database are extended to account for non-Oracle systems. This extension means that Oracle will determine the best method for executing the SQL statement. Oracle might determine that the greatest distributed performance can be achieved by performing the JOIN of two tables in the non-Oracle system at the remote non-Oracle site. In this case, only the rows that satisfy the SELECT statement will be returned to the originating Oracle site, which may greatly reduce the amount of data transmitted via the network. Additionally, collocated inline views can greatly improve the performance of a distributed query by accessing multiple tables of a non-Oracle system at once (reducing number of round trips).

For example, if the following SELECT statement was issued at a local Oracle site:

```
SELECT l.a, l.b, r1.c, r1.d, r1.e, r2.b, r2.c
FROM remote r1, remote r2, local l
WHERE r1.c = r2.c AND r1.e > 300 AND r1.c = l.c
```

Oracle may rewrite the SELECT statement and send the following to the gateway:

```
SELECT r1.c, r1.d, r1.e, r2.b, r2.c
FROM remote r1, remote r2
WHERE r1.c = r2.c AND r1.e > 300
```

Oracle Database Gateways

The capabilities, SQL mappings, datatype conversions, and interface to the remote non-Oracle system are the responsibility of the gateway. It interacts with Heterogeneous Services to provide the transparent connectivity between Oracle and non-Oracle systems. As your heterogeneous environment grows, you can add additional gateways to connect to any number of additional vendors' non-Oracle systems.

Oracle offers tailored gateways for many systems, DB2, Sybase, Informix, SQL Server, IMS, VSAM, Adabas, to name a few. These are specifically coded for the target non-Oracle system. They provide an optimized solution and are also end-to-end certified.

The gateway also provides the advantage that it can be installed independently of the Oracle database.

Database Gateway for ODBC

Oracle also offers a generic gateway, Database Gateway for ODBC. It is a generic solution that uses an ODBC driver to access any ODBC compliant non-Oracle system. It addresses the needs of data access to many data stores for which Oracle does not have a tailored solution. Database Gateway for ODBC makes it possible to integrate with low-end data stores such as MySQL, Foxpro, Access, dBase and non-relational targets like Excel. It is a feature of the Oracle database.

Database Gateway for ODBC Limitations

- BLOB/CLOB data cannot be read through *pass-through* queries.
- Updates or deletes that include unsupported functions within a WHERE clause are not allowed.
- Stored procedures are not supported.
- Cannot participate in distributed transactions—they support single-site transactions only.

Gateways are an integral component of the overall heterogeneous connectivity solution. Specifically, they provide:

- SQL and Data Dictionary Translation information
- Datatype Translation

SQL and Data Dictionary Translation Information

As previously discussed, the gateways contain all of the necessary translation information that the Heterogeneous Services translation service requires (see "SQL Service" for additional information). When a gateway is registered with the Oracle database, the SQL and data dictionary translation information is read from the gateway and is stored in the Oracle database.

Since this information can be stored locally in the Oracle database, this translation information is read only once from the gateway, which reduces the number of costly round trips.

Datatype Translation

Unlike the SQL and data dictionary translations that are performed by Heterogeneous Services in the Oracle database, datatype translations are performed by the gateway. There are two reasons for this:

- overhead reduction
- conversion of unsupported datatypes

For example, with the numeric datatype there are so many different types of numeric datatypes, performing the datatype conversion on the gateway reduces the overhead between the Oracle database and gateway. This approach eliminates the need to first transfer the many available datatypes to the database and then perform the translation to an Oracle NUMERIC datatype.

Datatype translation is expensive and by performing such translations on the gateway, the Oracle database will not be overly "taxed" by such an expensive operation, yielding greater database performance.

Finally, performing the datatype translation on the gateway provides the ability for the gateway to convert unsupported datatypes to a datatype that Oracle supports. (Note, however, that not all gateways support this feature.)

Gateway Mobility

Since the gateway is external to the Oracle and non-Oracle systems, there is some flexibility in where the gateway can be installed. The Oracle Database Gateway can be located with:

- the non-Oracle system
- the Oracle system
- a stand-alone system

Several issues must be considered when determining where to install the Oracle Database Gateway: network traffic and operating system platform availability, hardware resources, and storage.

If there are multiple Oracle databases that need to access one non-Oracle system then it is optimal to install the Database gateway on the computer hosting the non-Oracle database, as this will result in the fewest network connections.

If, however, a gateway is not available for a desired platform or the Oracle/non-Oracle database does not have the hardware resources or storage capacity for the gateway, you may install the gateway on a stand-alone computer running on an available platform for the desired Oracle Database Gateway. While this

configuration yields an extra network connection, which may result in poorer performance, this flexibility enables you to connect to the non-Oracle system.

HETEROGENEOUS TRANSPARENCY

Achieving heterogeneous transparency not only provides your users with a consistent interface to both Oracle and non-Oracle systems, it provides a stable foundation for your database applications to be developed upon. Instead of requiring applications to interoperate with non-Oracle systems using their native interfaces (which can result in intensive application-side processing), applications can be built upon a consistent Oracle interface.

Consider the scenario where a customer uses Oracle Database Gateways to access heterogeneously stored data with the plan to migrate the heterogeneous data to an Oracle system. If a database application is developed to interoperate with both an Oracle and a non-Oracle system using their native interfaces, once the migration to a homogeneous Oracle environment is complete, the database application would need to be altered to operate in that environment.

Operational Transparency

The target non-Oracle system appears as a remote Oracle system since the gateway provides the ability to translate SQL, data dictionary, and datatypes, to the proper dialect of the target non-Oracle system and also has the ability to securely manage the transactions with a non-Oracle system. This operational transparency provides a consistent interface for database application developers, which decreases development time and increases the mobility of the database application.

Location Transparency

In some cases, achieving operational transparency is not enough; sometimes it is advantageous to make it appear as though all remote objects are local. Location transparency can be achieved by defining a VIEW or SYNONYM for each remote object. Instead of referencing a remote object by specifying a database link, you can reference the VIEW or SYNONYM directly. For example, a user might execute the following SELECT statement to access a remote object:

```
SELECT empid, ename, mgr FROM
emp@non_Oracle_system;
```

After you create the SYNONYM emp on your local database, the user would access the remote object by executing the following SELECT statement:

```
SELECT empid, ename, mgr FROM emp;
```

In addition to eliminating the need to specify a database link to access a remote object, location transparency further increases the mobility of your database application. If the location of your remote database object changes, you simply

change the definition of the VIEW and/or SYNONYM and your database application remains unaffected.

CONCLUSION

Oracle offers Oracle Database Gateways for synchronous information integration. In a heterogeneously distributed environment, gateways make it possible to integrate with any number of non-Oracle systems from an Oracle application. They enable integration with data stores such as DB2, SQL Server, VSAM and Excel, transaction managers like CICS and message queuing systems like WebSphere MQ.

White Paper Title: Oracle Database Gateways
Date: July 2007
Author:
Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com