

## Oracle Database Links to SQL Server Simple Architecture. Complicated Solution.

**FTP, MQ Series, XML and Middleware are all examples of different ways of transporting data from one point to another. How much time and energy are spent implementing these solutions and at what cost? How many failure points are introduced as a result? Here are some things to consider when using database links and PL/SQL to access SQL server databases.**

As Windows-based PCs are becoming more and more a reliable and economic solution, Oracle and SQL Server-based applications are becoming more of a norm. As these databases become more robust and a greater number of applications have a need to integrate, table-to-table becomes a much more usable and reliable solution. It requires less time to configure, and offers fewer failure points, and more reliable integration.

Along with its benefits, there are a few pitfalls when creating Oracle-to-SQL Server DB Links. Creating the link is very well documented on the Oracle Website; however it is cumbersome to use. Here are some things to consider.

### PL/SQL Transaction Processing and Two-Phased Commits

Within a PL/SQL function or procedure, Oracle does not allow a transaction where a table over a DB link is accessed and then a local table is accessed. PL/SQL will indicate a compile error.

The first compile error is "ORA-02047 cannot join the distributed transaction in progress." This seems to occur when attempting to update an Oracle Version 6 or Oracle Version 7 database in the same transaction. This is caused by a transaction being in progress against a remote database that does not support two-phase commit. To resolve the error, the current transaction must be committed before attempting the action that caused the error. In other words, if there are any transactions pending before the database link, they must be committed before the call to the table over the database link.

If there is still a transaction, then there will have to be another commit to complete the transaction. Thus, a minimum of a two-commit transaction must be used when accessing both local Oracle tables, as well as remote SQL Server tables in a PL/SQL statement. This may lead to an increase in execution time as the number of records being processed grows.

## Field Definitions

Oracle-to-SQL Server DB Links suffer from field compatibility issues when certain field types are used. For example, VARCHAR fields in SQL Server do not seem to map across to Oracle. This is an issue when a field is implicitly named within a PL/SQL statement. However, CHAR fields map across just fine. Integer and small integer fields were also able to be mapped. There are many tools on both Oracle and SQL Server to handle dates. Date fields do not map either.

## Oracle User

The Oracle user will need to exist on the SQL Server side with the same password that exists within Oracle.

## PL/SQL Cursors

Tables over a database link cannot be called within a PL/SQL Cursor. There were no compilation errors, but at runtime, there are issues. One error is "ORA-14552: Cannot perform a DDL, commit or rollback inside a query or DML." This was caused by no data existing in the remote database table that

was being updated. Another error that can be thrown is "ORA-01002 fetch out of sequence." This was occurring when fetching from a SELECT FOR UPDATE cursor after a commit.

A PL/SQL cursor loop implicitly issues a FETCH statement and may also cause this error. To resolve this error, the 'COMMIT' statement was removed from the procedure. After working through these errors, eventually an "ORA-01010 invalid OCI operation" and/or "ORA-02063: preceding 2 lines from <DBLINK>" errors were displayed. This is where the feasibility of utilizing a cursor comes to an end. At this point, the only feasible approach is to loop and select the data that you need from the table on the other end of the DB link.

***Ben Sommer is a senior consultant and Varun Arya is a developer at enVista. For more information, contact enVista at 877-684-7700 or at [inforequest@envistacorp.com](mailto:inforequest@envistacorp.com).***