I am often asked by customers, “How can I make my application talk to both Ingres AND another database”?

Usually, customers want to do this because either:

They are writing a new application for Ingres that needs to access legacy information from another system.

or

They are migrating an application to Ingres, that was previously written for another database, but they still need to work in conjunction with that other database.

This article discusses the different scenarios available with Ingres for applications to work with multiple database managers. This article is very high-level and does not contain all important details. In reading this, some knowledge of transaction management is helpful, but not required.

The three scenarios include:

Application Transaction Control via Individual Drivers

Application Transaction Control via Ingres Distributed Architecture and Gateways

Application Transaction Control via a Transaction Manager
1. Application Transaction Control via Individual Drivers

The diagram below illustrates an application that communicates with Ingres in addition to another database manager. In this configuration, the application, not the database manager, assumes responsibility for the completion and integrity of the "logical transaction". The application maintains separate connections to each of the databases, using each of the databases’ appropriate drivers.

For example: for an ODBC application that communicates with Ingres and Oracle, you would open an Ingres ODBC connection and you would open an Oracle ODBC connection. You would need the appropriate ODBC drivers for both Ingres and Oracle and have the appropriate licensing for both systems. The application would operate upon either connection separately.

In this scenario, if you have an INSERT/UPDATE/DELETE transaction that logically rolls back, it would be your application’s responsibility to facilitate the rollback on each database. Likewise, if one of database managers instigates its own rollback, it would be your application’s responsibility to detect this condition and instigate a corresponding roll back on the other database. This would not happen automatically.

Note: for read-only operations, rollback may be a non-issue depending on the read-level locking you have implemented and system isolation. For systems using high-levels of isolation, you might still need to manually instigate a rollback simply to release read locks.
2. **Application Transaction Control via Ingres Distributed Architecture and Gateways**

The diagram below illustrates an application that communicates with an Ingres database. The application is unaware of the actual location of data and is simply written to a singular Ingres interface, which takes care of transaction management.

Underneath the covers, Ingres, uses Star (the Ingres distributed database technology component included with Ingres Database) and in turn, one of our gateway products, Enterprise Access EA, or Enterprise Database Connectivity (EDBC), to communicate to the other database managers.

Enterprise Access (EA) is used as the gateway for Oracle, SQL Server, and DB2 UDB. EDBC is used for IDMS, DB2 z/os, IMS, VSAM, and Datacom. Both EA and EDBC provide read and/or write access to other databases and both products require the appropriate drivers and licensing to communicate to those other databases.

For example, you might have a .NET application that works with data from both Ingres and DB2 z/os. The .NET application, running on a Windows platform, would simply use the .NET managed provider included with Ingres. The application would be written as if to a single data source, Ingres, and would rely on Ingres to handle all the detailed transaction control.

The Ingres database would appear to contain all applicable tables, but under the covers, in addition to Ingres tables, there would simply be pointers to a gateway, in this case, EDBC. EDBC would be setup with the appropriate drivers for mainframe access, which in the case of DB2 z/os, requires an agent on the mainframe.
Your application would still statements like SELECT FROM table, but this table name would really be a synonym for a table accessed automatically through the gateway with Ingres Star and EDBC.

It should be noted that there are some considerations associated with using gateways. First, because of the nature of gateways using non-Ingres technology, the error messages returned from statements executed through gateways may be different than directly executed statements. So, if you move an existing direct-to-Ingres application that checks for specific error codes to a gateway implementation, you will need to thoroughly review error codes. Second, the breadth of SQL supported by the gateways is likely to be a subset of SQL supported by a direct Ingres connection. This is because either the target technology does not support all Ingres SQL statements or because of limitations of the gateways themselves. A detailed discussion of these considerations is beyond the scope of this paper, but you should be aware of them and keep them in mind.

3. Application Transaction Control via a Transaction Manager

The diagram below illustrates an application that uses a third-party transaction manager, TM, to control distributed transactions across database products. Transaction manager products include Tuxedo, JBOSS, CICS, and Encina. The application makes function calls to the TM (via the TM’s own API). The TM is then responsible for connection and interface to each of the databases, via the appropriate drivers and licensing. In order for transactions to span multiple databases in this environment, you must use the industry-standard TP/XA protocol. Ingres Database packaging includes support of TP/XA.
Application Server Considerations

I should add a few thoughts on application servers as they are very popular technology. Application servers have many benefits including the abilities to horizontally scale applications and standardize application interfaces.

Applications servers themselves cannot facilitate distributed transactions, but they can, however, be inserted into any of the above scenarios.

a) Application Transaction Control via Individual Drivers

Your application can communicate with an application server which in turn uses individual drivers.

b) Application Transaction Control via Ingres Distributed Architecture and Gateways

Your application can communicate with an application server which in turn uses gateways.

c) Application Transaction Control via a Transaction Manager

Your application can communicate with an application server which in turn uses a transaction manager.

For more information on any of our products or services please visit us on the Web at: www.ingres.com