Comparing Three Leading DBMS Vendors' Approaches to Replication.

by Charles Thompson

Database Replication

The promise of distributed databases has been with us a long time. Until relatively recently, though, the technology available to set up a distributed database was either too limited or too complicated. Today, database replication, the copying and maintenance of data on multiple servers, has come of age. Every major database vendor has a replication solution of one kind or another, and many nondatabase vendors also offer alternative methods for replicating data.

In this article, I look at three of the top databases and compare their replication technology. For various reasons, I have chosen to look at replication solutions from Microsoft Corp. (Redmond, Wash.), Oracle Corp. (Redwood Shores, Calif.), and Sybase Inc. (Emeryville, Calif). Other database products, including those from Informix Software Inc. (Menlo Park, Calif.), IBM Corp. (Armonk, N.Y.), and Computer Associates International Inc. (Islandia, N.Y.), have advanced replication technology, and you can build solid replication architectures with middleware products such as those from Tuxedo and TopEnd. Before I look at descriptions of the three different approaches, though, it would be worthwhile to discuss some of the issues with replication.

As I look at more replication products, I keep thinking the same thing: This is not a toy to be taken lightly. Don't be fooled by the easy configuration of some products, notably Microsoft's -- replication is serious business. Even if the user interface simplifies the implementation and administration, replication requires careful analysis and planning. It is also important to have a thorough understanding of the mechanism that your database uses for replication.

Managing a distributed database is vastly more difficult than managing a centralized database, and time spent justifying and planning for the extra work is worthwhile. Write down your reasons for setting up a distributed database. Draw a diagram of your database network, then define exactly which data you wish to replicate and which form it will take -- will it be a full copy, subset, summary, or combination of source data? Use a word processor, spreadsheet, drawing program, or CASE tool to document your requirements. This should give you a good idea of the task ahead.

If you are replicating in order to have a "hot standby" system, look into alternatives to replication, such as hardware solutions and database mirroring. In such situations, it pays to use a system that was built for hot standby -- some of the issues are different from those for regular data distribution.

It is extremely important to plan for all or part of your network going down. Often, this is the main reason for replicating in the first place -- remote users can continue working even if the connection to the central site is down. Some replication systems automatically synchronize all replicas when the network is available again, and some require manual intervention. You will need to decide whether your applications will have to change in order to take advantage of this.

Finally, make sure that all database administration takes the distributed database into account. This is especially important in an environment where the staff is familiar with a centralized database. Decisions regarding database manipulation, recovery, and network management are different when the data is distributed.

Replication Scenarios

To compare the replication products, I will use three different replication scenarios: simple read-only replication

(as illustrated in Figure 1), replication to and from a mobile client (as illustrated in Figure 2), and multiple updates. These are representative of most distributed architectures. They may be used in a variety of architectures, for systems that provide:

- data distribution to a network of servers, including those that are mobile or occasionally connected (illustrated in scenarios 1 and 2)
- data consolidation to a central server (scenarios 1 and 2)
- process separation onto more than one server (scenario 1)
- information flow from one server to another (scenario 3)
- data sharing across multiple sites (scenario 3)

Throughout these examples, I will use the concept of a three-tier physical server architecture, which includes a central server and two workgroup servers (the client is the third tier). In a real situation, there could be more than one central server (including, for example, a data warehouse server) and many workgroup servers. A workgroup server might support part of an office (such as a single department) or more than one office. The network might include local and wide-area connections.

With read-only replication, the data is entered and stored on each workgroup server. Only data relevant to each local workgroup is located here. The data is also replicated to the central server (consolidated), so the central server will contain a read-only copy of all data from all workgroups. The data on the central server (logically stored in a single table) has two sources, but each individual record only has one source.

Alternatives to this scenario, but essentially the same, are: 1) the data is entered on the central server and copied to the workgroup servers, where it is read-only; or 2) the data is entered on one workgroup server and replicated to one or more other servers as read-only copies.

Mobile computing has become much more accessible in recent years, and in most organizations some people work away from the office. There are now a number of methods for providing data to a mobile workforce, one of which is replication. In this case, the data is downloaded on demand from a local workgroup server. Updates to the workgroup or central data from the mobile client, such as new customer or order information, are handled in a similar manner.

Multiple updates are the classic nightmare of replication architects. Sometimes it would be so much simpler to let any server update any of the data and have it automatically replicated to all other servers. The problem, of course, arises with the possibility of two updates being made to the same data on different servers. The traditional solution is to designate a single source server -- that way, you would have an architecture similar to our first scenario. The latest sophisticated replication technology, however, enables this architecture and makes it work by allowing the administrator to make some arbitrary decisions about which data is "correct." Any conflicts are processed by a conflict-resolution algorithm that usually chooses one of the following methods:

Priority: each server is given a unique priority; higher-priority servers "win" over lower-priority servers
Timestamp: the latest or earliest transaction in the conflict is considered correct; if you choose to do nothing in a conflict situation, the latest transaction wins
Data partitioning: each row is guaranteed to be manipulated by only one server; this simplifies the architecture to our first scenario

Many other conflict-resolution or avoidance schemes can be implemented by the various products.

Another major problem with replication in general has to do with automatically generated keys. Most systems now use an automatically generated surrogate key to uniquely identify a row of data. When data for one table is created on a number of servers, you need to ensure that the replicated rows are unique. There are three accepted ways to do this:

1. set up the key generators to use a predefined number range, different for each server
2. add a server identifier to the primary key
3. replicate into separate tables and read all of the data through a union view; to deal with the potential of duplicate keys in the UNION, a pseudocolumn would be used to represent the source database

Microsoft SQL Server 6.5

It's no secret that Microsoft's database technology is derived from that of Sybase. Microsoft's first version was nearly identical to Sybase's SQL Server, but the two databases have since gone separate ways. The two companies developed replication technology independently, leading to replication mechanisms that are quite different.

SQL Server 6.5 runs on NT only, but it has a large market share in that environment. It is also becoming more and more popular as NT is chosen for an increasing number of operating system installations. In keeping with the ease of use that is Microsoft's Fort?, the replication in 6.5 is straightforward and simple to set up, and it virtually runs itself.

Publish and Subscribe

Microsoft uses a "publish-and-subscribe" metaphor: The DBA makes the data on one database (the publisher) available to the world, and then another database (the subscriber) receives the replicated data. In keeping with this metaphor, the machine that handles the job of moving the data from publisher to subscriber is called the distributor.

As with all distributed databases, the replication environment must be designed carefully. With Microsoft, you need to include the following elements in your architecture: the data to be replicated (publications), the database carrying the source data (publisher), the database receiving the replica (subscriber), the distribution machine, how you want the subscriber to use the data (read or update), and how often you need the data to be replicated.

Once you have a replication design, you will find it quite easy to set up your servers. You use the same user interface -- SQL Enterprise Manager -- that you use for other administration tasks, and you designate the publisher database. You then define the publications and select the subscribers.

This simple setup can be used to configure complex replication architectures. It is easy to use the basic tools to build up a structure that can be very difficult to maintain. Any server can be both publisher and subscriber (and distributor, for that matter), and the same table can contain replicated and source data! It will take a bit of self-control to figure out how to make that work.

SQL Enterprise Manager has a unique graphical view of your replication network (as shown in Figure 3), showing the topology as a network rather than a hierarchy, as most other tools do. You can move and place your publishers, subscribers, and distributors on the workspace in any configuration you like.

With this tool, it is relatively easy to set up complex configurations, but the database is not always robust enough to manage them correctly. While I was trying out a variety of options for this article, the server crashed or hung up on me a number of times, even after I reinstalled the whole database from scratch. I don't know how common this problem is, but I know my colleagues in another city had similar problems on their server. Also, the DBA should think through the architecture very carefully, because the system will let you do nearly anything, whether you really want to or not.

The Mechanism

Each RDBMS uses a different mechanism to manage the replication. Microsoft's solution is to use the existing SQL Executive Task Manager. This is a module of SQL Server that lets you run scheduled or on-demand DBA tasks. When you set up a replication architecture, tasks are created and scheduled on the distribution server. These perform the actual replication by reading the logs on the publisher, logging on to each subscriber, and implementing all changes from the publisher by issuing the appropriate SQL commands. With multiple subscribers, keeping these tasks in sync with your architecture can become very difficult.

Case 1: Read-Only Replication. This first example is very simple to set up and manage. You publish table A on each workgroup server, and then you subscribe to the workgroup servers on the central server. You can put the replicated data from all workgroups into a single table, so you should make sure that the data coming in from the workgroups is unique so that you don't have conflicts.

Case 2: Mobile Data. Microsoft has an incomplete solution for mobile databases: Use the Jet engine (from Access or Visual Basic) as the mobile database. Microsoft lets you set up limited replication through ODBC to Jet. This is a new facility in 6.5, and it does not allow full replication. The two databases (SQL Server and Jet) use different versions of SQL, and Jet has fewer capabilities, so your mobile application needs to be quite different

from your nonmobile application. Basic one-way replication of small amounts of data may work for you, but if you want a more complex distributed database, you may want to consider writing the replication routines into mobile applications.

Case 3: Multiple Updates. SQL Server permits multiple subscriptions into a single table. The only mechanism you have for conflict resolution is the triggers on the target table. The same table can be subscribed to and published. The mechanism is fairly simple but provides few safeguards, so it is up to you to think through all scenarios carefully and write the appropriate trigger code to manage the data correctly.

Microsoft's approach to publish-and-subscribe replication technology helps make this product easy to set up and maintain. The mechanism is also flexible enough to build a very complex distribution architecture, and the tools provide a clean, consistent enough interface to manage this effectively. Limitations such as the restriction to the NT platform only will limit this product's appeal, and Microsoft has yet to make SQL Server as robust as the other leading RDBMS products.

Oracle 7 Release 7.3

The Oracle Server comes in various flavors, including Universal Server, Parallel Server, Enterprise Server, and Workgroup Server. (For information on Oracle 8 and its support for replication, see Clara Parkes's interview with Oracle's Jerry Held and Ken Jacobs on page 86.) They all contain the same core database server functions but include different options and run on selected groups of platforms. For replication, I am interested in the Enterprise and Workgroup servers. There are three important distinctions between these two products:

1. **Price:** Workgroup is significantly cheaper.
2. **Supported platforms:** Enterprise is targeted at medium- to large-scale servers, and Workgroup is targeted at smaller servers, but there is some overlap, notably on NT platforms.
3. **Replication:** Advanced Replication is only available on Enterprise Server.

Oracle provides a number of mechanisms to develop replication architectures, including transparent two-phase commit, snapshots of various forms, and advanced replication.

Two-phase commit is a mechanism you can use when you need synchronous (realtime) replication. When you set up a database link and use it in a transaction that updates two databases, Oracle's two-phase commit transparently makes sure the transaction completes successfully (or rolls back the completed part of transaction if it is not successful). The inherent complexity of designing realtime transactions across multiple databases led to the development of other replication methods that will do most of the work for you, but in batches (asynchronously). Two-phase commit is used as a basic building block in these other forms of replication.

Snapshots

The snapshot is the original replication mechanism in Oracle and now has many options. A snapshot can be "fast" (only logged changes are replicated) or "complete" (the whole table is copied). It can be read-only or updateable. The latter can be useful if you wish to allow temporary modifications at a remote site that are valid until the changes are "approved" at the central site.

The difficulty with the snapshot is that it is not all that robust. For a simple, small distributed system, it can be quite adequate, but the more complex your requirements, and the more important your replicated data is to you, the more you need Advanced Replication.

The major limitation with snapshots is that if they do not complete, they will only retry up to 16 times. After that, you must restart them manually.

Snapshots are also limited in their capabilities. A fast snapshot, for example, must be based on a single table; if you need a multitable join in your snapshot, you must use complete snapshots. A fast snapshot also requires you to define a snapshot log on the source server and define the snapshot on the target.

When a snapshot is running (at the time you configure), it executes in a separate operating-system process. Because of the naming convention for these processes, you may only have 10 snapshots running at any one time.

Advanced Replication

The latest replication technology from Oracle is only available with the Enterprise Server. Advanced Replication uses a mechanism based on system-level stored procedures (running as "jobs"), triggers, queues, and database links. With this system, you can configure exactly what gets replicated: complete tables or changes only. You can also program your own conflict-resolution algorithm. Replication can be synchronous or asynchronous and can be swapped between the two as your needs change.

The beauty of this mechanism is that it uses basic Oracle server technology, so it is well integrated into the server. It can be configured and managed with your favorite SQL interpreter or DBA tool, because the commands are normal SQL and PL/SQL procedure calls.

The best way to manage your Advanced Replication, though, is with Replication Manager, a component of the Oracle Enterprise Manager (widely available only since mid-1996). This is a comprehensive toolset for server design, configuration, and management, but it also extends to software management. Replication Manager lets you create replication groups, which makes a large, complex distributed network much more manageable. It uses a familiar drag-and-drop, graphical interface, much like other graphical DBA tools.

Case 1: Read-Only Replication. Snapshots were originally designed for this scenario. Snapshots are simple to design and configure for this kind of architecture. If you only have Oracle Workgroup Server, this is the ideal way to build your replication architecture. Advanced Replication will also work very well here, adding extra flexibility and robustness to your replication. This eases the management of a large number of replicas, lets you define complex subsets and joins from the source data, and improves the system's handling of intermittent connections or network outages.

Case 2: Mobile Client. Oracle has a number of solutions for the mobile client, but not one of them is completely adequate. You can use the following products:

- Personal Oracle7, a single-user version of the server with all of the features of the Enterprise Server
- Oracle Lite, a much smaller standalone database without all of the server capabilities (missing mainly PL/SQL, the programming language for triggers and stored procedures)
- any other database with Oracle Mobile Agents

Personal Oracle7 is a very large application (more than 100MB disk space required) that, although it has all of the normal server capabilities, is nearly unsuitable for any real applications. Why? Well, it requires a very powerful PC (a Pentium-100 is a bare minimum) with at least 24MB of RAM, and even then, performance is barely adequate. Its big advantage is that you can use your full client/server application on the mobile client without any changes. Replication can use snapshots when attached to the network, or you can build a replication-on-demand module into your application, using two-phase commit.

Oracle Lite is a typical slim database (it takes up less than 1MB of memory) that was designed to take up a minimal amount of system resources in a Windows environment. It looks and feels like an Oracle database but misses out on some standard features, such as the PL/SQL programming language. It also has some extra capabilities, such as an "object-relational" approach (making it easier to use with object-oriented tools such as C++). To use it, you will need to either limit the capabilities of your database (so that both the central and mobile databases are the same) or write a specialty application just for mobile users.

Mobile Agents is a new technology for configuring asynchronous tasks for mobile PCs. It consists of an API and process on the client, with a coordinating process on your server. The problem is that you still need to write your own code in order to use it, which can be a complex task.

Case 3: Multiple Updates. Advanced Replication was made especially for this and much more complex architectures. The toolset lets you manage the architecture in a reasonable way and provides more than enough capabilities for your requirements (who is going to use more than one or two of the 10 built-in conflict-resolution schemes?).

Oracle is the leading database company, and it is easy to see why its database server is the company's leading product. Oracle has covered nearly every base with some product feature, and it lets you build a complex replication architecture that is very robust. With such a variety of options and features, you'll need to be aware of the particular combination of features in your product (Enterprise or Workgroup Server). If you wish to make your

database application available to your mobile users, you'll need to prepare for a significant amount of extra programming.

Sybase SQL Server System 11

Sybase has built yet another way to replicate data, using a separate product, Replication Server, to manage replication for you. Replication Server is a separate set of processes with its own command language (although you can access its commands through the ISQL interpreter, the same tool you can use to administer the database server). Sybase's replication has been available since 1993, longer than either Microsoft's or Oracle's.

Replication Server uses a publish-and-subscribe metaphor, much like Microsoft. Database objects are defined as publications on the source server, and they are subscribed to by the target server(s).

An interesting thing about Sybase is that it lets you replicate stored procedure calls. This is elegant -- and potentially a big bonus in performance -- because a stored procedure call will execute the same way on a number of machines, cutting the expense of sending potentially thousands of rows across the network. Microsoft and Oracle can do this as well, using a dummy table with a trigger on it to run the stored procedure. Sybase, however, replicates stored procedures explicitly and doesn't require any trickery.

To design your replication architecture, use another graphical tool called Replication Administrator. This one is similar to Oracle's but manages the Replication Server, which can run on one or more servers.

The mechanism used by Replication Server is similar to Microsoft's but uses specialty operating-system processes. In addition to the Replication Server process, you need a Replication Agent at each publisher. This reads the write-ahead log files and passes relevant data changes on to the Replication Server, which then sends the changes to the subscribers.

The Replication Agent uses Sybase's standard open connectivity, so a wide variety of these are available for non-Sybase databases. This is a distinct advantage over Oracle, which relies heavily on PL/SQL, available only on Oracle databases.

On the other hand, Sybase does not provide easy synchronous replication -- this must be done using your own code and a two-phase commit.

The Replication Server lets you employ complex store-and-forward architectures by using multiple replication servers in a chain. One server can publish to another replication server, which can publish on to more servers or target databases.

Case 1: Read-Only Replication. This is a fairly simple architecture to set up. As with Microsoft, you publish table A on each workgroup server and subscribe to each copy of table A on the central server. With a large network of workgroup servers, this process could be quite difficult to manage, so you might want to consider using multiple Replication Servers, each managing a small group of workgroups.

Case 2: Mobile Client. Sybase is the only system of the three that has a comprehensive solution for mobile computing. SQL Anywhere, the old Watcom database, is now fully compatible, even down to Transact-SQL, with SQL Server. This enables your application to run virtually unchanged in the mobile environment, saving a significant amount of development effort. To replicate between SQL Anywhere and another server, on an intermittent or on-demand connection, use SQL Remote. It is not as complete or flexible as Replication Server, but it provides a basic level of replication to support most mobile applications without special programming.

Case 3: Multiple Updates. Replication Server manages this type of data distribution fairly easily and gives you complete control over the conflict-resolution algorithm by allowing you to select a built-in conflict-resolution scheme or program your own.

Another publish-and-subscribe server, Sybase Replication Server, lets you replicate both tables and stored procedures. It also manages replication between Sybase database servers and those from other vendors. One or more replication servers, which may be independent from the database servers, may replicate to each other, potentially simplifying the management of a complex architecture. Sybase also has a satisfactory solution for mobile computing that lets you use the same full application for both permanent and mobile network clients.

Practical Upshot

I have covered a number of aspects of three very different approaches to data replication. Microsoft and Sybase use a publish-and-subscribe metaphor, which makes the design and configuration of a complex architecture relatively simple. Oracle provides a comprehensive facility, with a hierarchical approach to replication.

Microsoft's product may not be ideal for critical systems, because it is restricted only to NT platforms and lacks the robustness of the high-end databases. However, it works very well within the target market for SQL Server 6.5 (small companies or departmental systems) by providing an easy-to-use interface, simple procedures, and a high level of integration with the operating system. The offerings from Oracle and Sybase, however, could be used in the most advanced, complex, and important distributed architectures because of their attention to detail and robust construction. These two high-end database systems cover all important aspects of replication but are more difficult and complex to manage than Microsoft SQL Server.

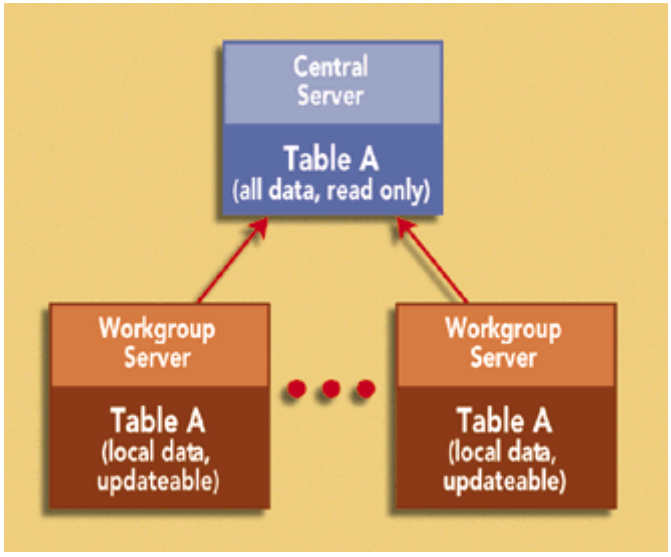
Regardless of the product you use, replicating data is not a simple task and requires a significant amount of planning and expertise in order to work properly. There are many potential pitfalls, and a good knowledge of the capabilities of your product will ease the way.

Charles Thompson is a manager at a large consulting firm in Wellington, New Zealand. He specializes in client/server design and deployment. You can contact Charles via DBMS at DBMS@mfi.com.

---

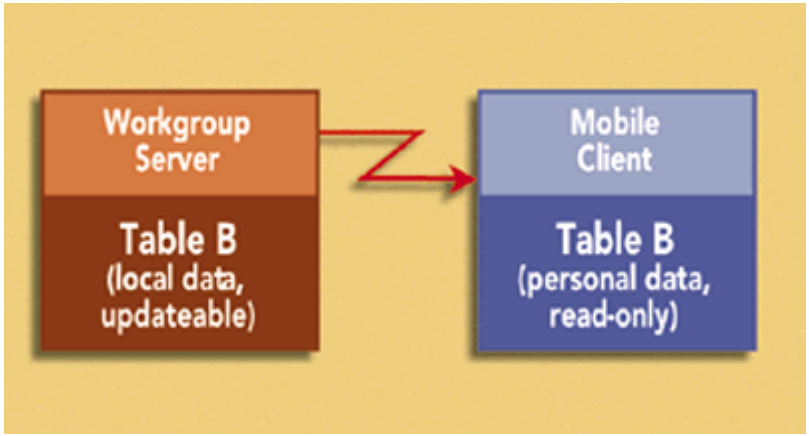
* Microsoft Corp., Redmond, WA; 800-426-9400, 205-882-8080, or fax 206-936-7329; www.microsoft.com.
* Oracle Corp., Redwood Shores, CA; 800-672-2537, 415-506-7000, or fax 415-506-7200; www.oracle.com.
* Sybase Inc., Emeryville, CA; 800-879-2273, 510-922-3500, or fax 510-922-9441; www.sybase.com.

Figure 1.



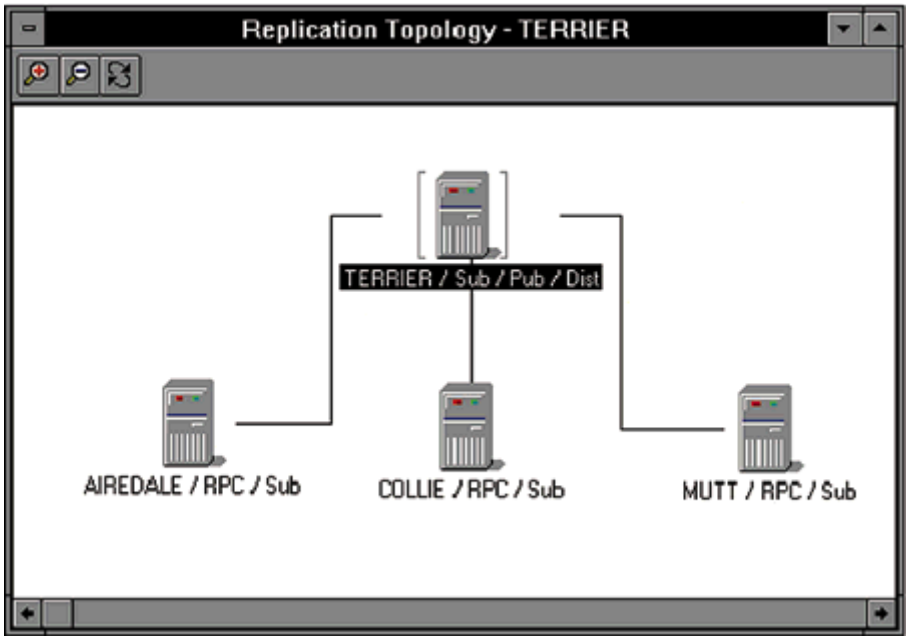
--An example of simple read-only replication.

Figure 2.



--Replication to and from a mobile client.

Figure 3.



--The Replication Topology screen in Microsoft SQL Enterprise Manager.

May 1997 Table of Contents | Other Contents | Article Index | Search | Site Index | Home

Please send questions or comments to dbms@mfi.com
Updated Monday, April 14, 1997