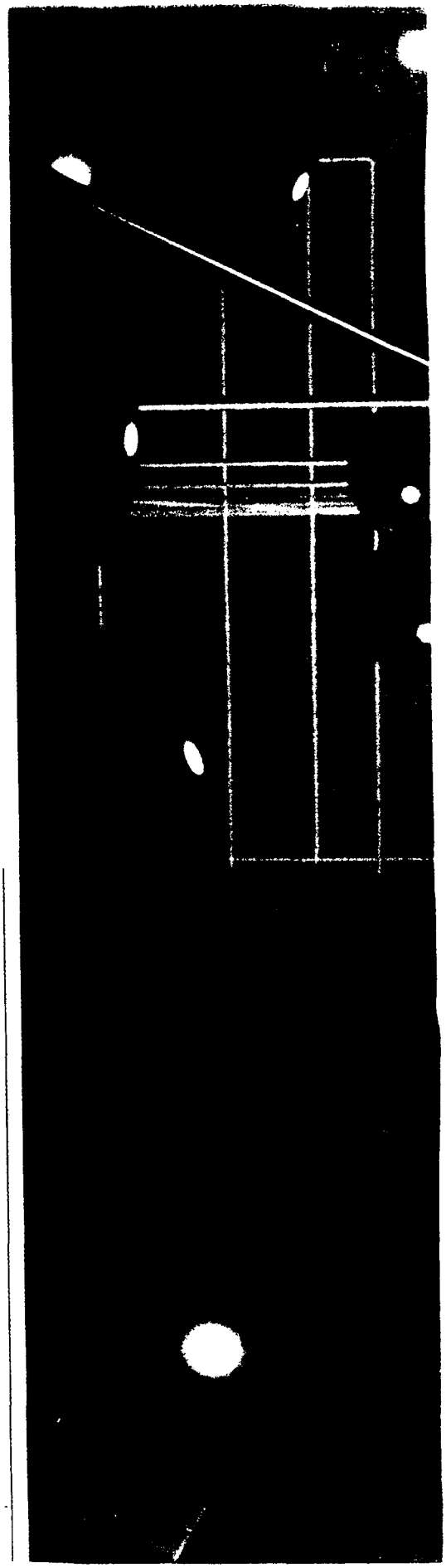
BY DOUG STACEY

Is replication salvation or the devil in disguise? Here's what three implementations tell us

Replication: DB2, Oracle, or Sybase?



REPLICATING DATA, propagating data, trying to manage copies of copies of data: The never-ending push toward giving users more direct access to data has become a major concern for many companies. You've heard of the new Information Superhighway? Welcome to the Information Junkyard! Continually fulfilling user requests for data by creating yet another data store on yet another local area network or open system and rigging up some way to get the data to it often results in redundant, hopelessly out of date copies of data. This article will review what leading vendors are doing to provide relief to beleaguered DBAs who are most often faced with managing the Information Junkyard.

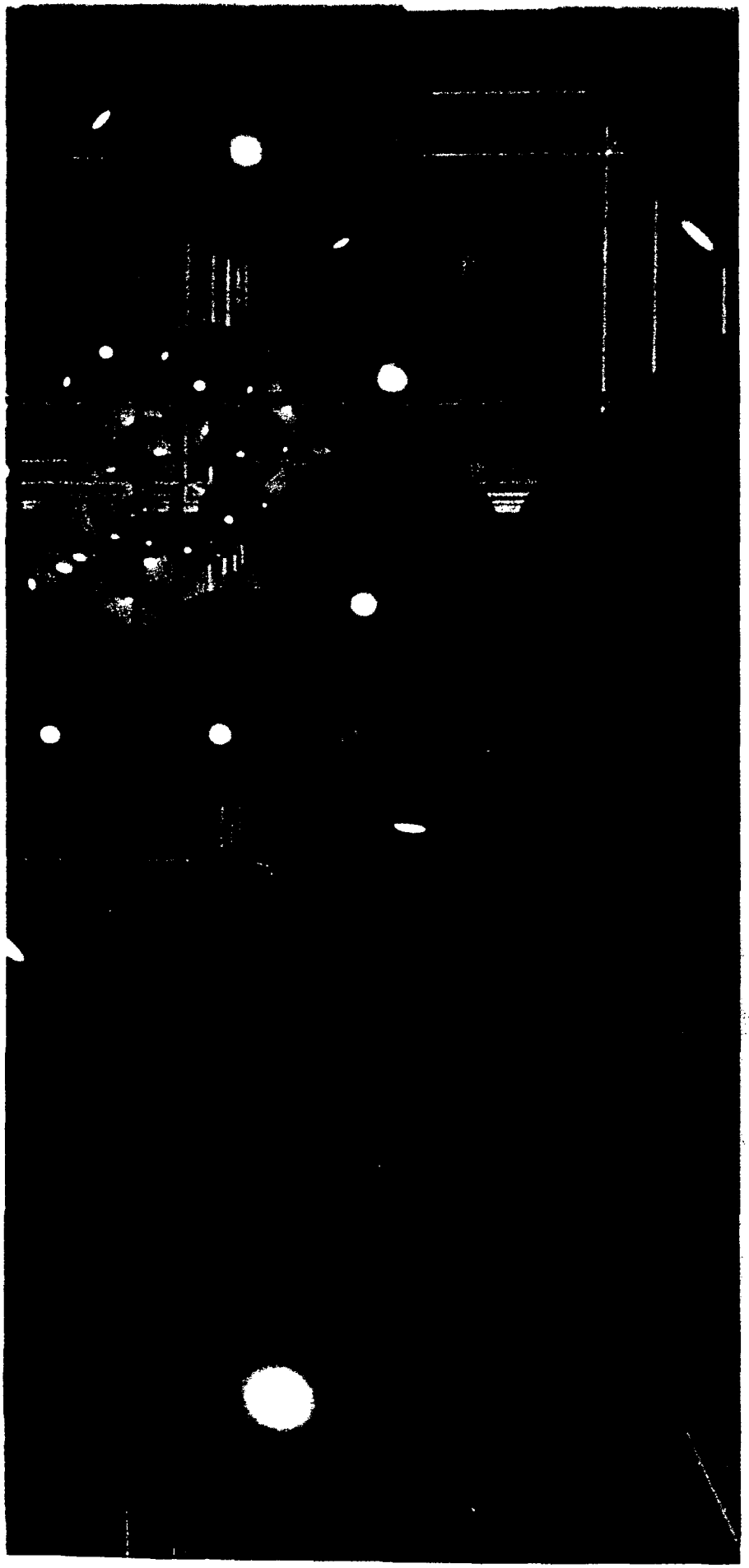
MOVE TO REPLICATION

In the late 1980s, distributed database processing was often touted as *the* answer to the problem of how to take advantage of the new platforms available to systems designers. The intention was simply to partition the data among locations to take advantage of local processing speeds and perform a distributed request to gather the data back together for any application requiring the global view.

This approach sent the database vendors scurrying to provide the various levels of distributed capability we see in their products today. It soon became clear, however, that this scenario of remote updates and distributed two-phase commits (2PCs) was not such a good idea. As the number of sites with data increases, imagine the response time problems in having users wait for their transactions to complete while the DBMS is busy coordinating commits between the remote sites.

The more popular solution has become *replication*: to copy the data and place it locally for processing. This trend has forced vendors to provide some means of facilitating the creation and management of the replicated data.

In this article, we'll take a close look at Sybase System 10's



Replication Server, IBM's Copy Management solution, and Oracle Corp.'s Oracle7 Symmetric Replication. After reviewing the major features and philosophies of each, we will compare the offerings.

Vendors must make a basic choice when designing a DBMS system to keep data copies current after initial replication. They can either:

☐ Update remote copies synchronously with the primary copy (that is, in the same unit of work [UOW] or commit scope), or

☐ Propagate the changes asynchronously, after the updates to the primary copy have been committed.

Synchronous replication (or tight consistency) provides the greatest level of data integrity. An update to the primary and one or more remote data copies is coordinated in the same UOW by performing a 2PC. Either the primary and all remote copies are updated together or all updates—including the one to the primary and the originating transaction—fail.

What's wrong with this approach? All replicated copies must be available any time an update is performed on the primary copy. As the number of replicated copies grows, the likelihood of one of the communications routes not being available increases. This problem can cause frequent transaction failures at the primary site since all updates to the primary copy will fail if any remote copy is temporarily unavailable.

Asynchronous replication (or loose consistency) rectifies these failures by first committing the update to the primary copy and then propagating to the replicated copies after the fact. If one of the replicated copies is unavailable, the update is stored for later retransmission to the unavailable site. The available replicated copies are updated as soon as possible, often within seconds.

The danger, however, is this: If an update that was committed on the primary copy is somehow *not* applied to a replicated copy, the two will become out of sync. Failures at the replicated copy could be due to a space problem or a general DBMS failure. Failures could also result from an application updating the replicated copy

ARTWORK BY: MICHAEL SCIULLO/TIB WEST©1994

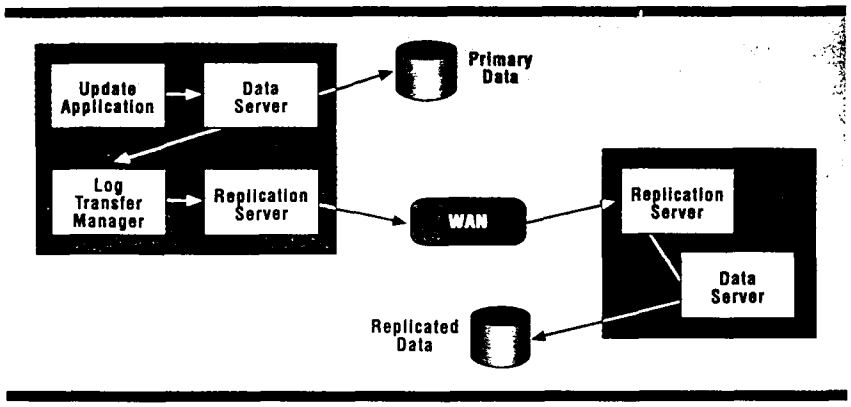


FIGURE 1. *Sybase Replication Server system components.*

locally, outside the replication system's control.

SYBASE REPLICATION SERVER

Many users considering the Replication Server have already rehosted an application (that is, moved an entire application from one host platform to another) or created a new application using Sybase SQL Server. The goal in replicating data in this environment is improving performance by moving data copies closer to users on local servers. Having to cross too many bridges or routers for data in a wide area network leaves most users with a less than satisfying work experience.

Sybase has chosen the loose consistency model for System 10 Replication Server. All updates are committed to the primary copy before being transmitted asynchronously to the replicated sites. Remote sites are created by using the Replication Command Language (RCL) to "subscribe" to that site for a data copy.

Two recently added components are integral to the Sybase replication system: the Log Transfer Manager (LTM) and the Replication Server (see Figure 1). The client application updates a data server just as any normal application would. (Data servers, by the way, do not have to be Sybase or even a relational databases; any system supporting a specified set of required data operations and transaction-processing directives can act as a data server.)

The LTM reads the SQL Server transaction log to look for changes to the primary data. Upon detecting a change, it notifies the local Replication Server. A Sybase LTM is provided with the system. If you are using a data server other than Sybase, it is up to you to provide an LTM for it. Every site that has a primary data copy for which replication will take place must have an LTM. LTMs are not required at a replicated site.

The Replication Server performs a lookup in the Replication Server system tables to determine which sites have subscribed for a copy of the changed primary data. Upon being notified by the LTM that a UOW has committed, it then communicates with Replication Servers at those sites, passing the necessary information for the update. The remote Replication Server then updates the replicated copy. A Replication Server must reside at both the primary and remote sites.

In this scenario, all updating must be performed at the primary site. Any application running at a replicated site that wants to update the data must either invoke a remote stored procedure at the primary site (which is done synchronously) or use a feature that's called Stored Procedures for Asynchronous Transactions.

Applications generate asynchronous transactions by executing special "replicated" stored procedures at the remote site. The procedure and its parameters are transmitted asynchronously to the server with the primary data copy and executed there. The updates then flow to the replicated copy through the normal mechanism. Note that an asynchronous transaction's failure causes it to be queued at the primary site. Manual intervention is needed to correct the problem and reexecute the transaction.

When subscribing for a replication, it is possible to use an optional WHERE clause that specifies which rows of the primary table should be replicated. In this way, a replicated site can obtain only the data in which it is interested. For instance, a particular region could subscribe with the primary site to obtain only the copies of orders that originated in their region. This approach would limit overall network traffic by reducing transmission of unnecessary data.

It is also possible to update a table in fragments. Consider a table that must be updated locally by each of several branches in a company. Each branch becomes the primary site for its fragment of the database. Each branch is also a replicated site for all other database fragments. Any updates done by a branch for its primary fragment are replicated to the remote sites.

IBM COPY MANAGEMENT

With the majority of data today still residing on mainframes, how do you fulfill the desire to allow users to access data from graphical user interface (GUI) programs (either for decision support or for transaction processing)? Typically you have three choices:

- ☐ Move data to a new platform, which requires that all applications using the data be rehosted to the new platform as well
- ☐ Use gateway technology to perform cross-system access back to the mainframe
- ☐ Create and maintain controlled data copies on new platforms.

The first approach, rehosting, can be extremely expensive. It often involves rewriting an application in a new language, using a new data store. Many who have tried this option have found that the target environments were not mature enough to handle moderate to large applications. If *all* the data moves to the new platform, all batch processing must move there as well. Rehosting has provided additional obstacles to overcome: job scheduling, tape management, not to mention handling the sheer power available on a single CPU.

The second option, gateway access, frequently causes performance problems. Today's gateways (based on LU6.2 or TCP/IP) must access

the mainframe each time data is required—and performance suffers. Gateway technology is sure to improve; but at this point, gateways are best for casual access.

Which leaves us with the third option: duplicating data from one platform to another. While creating and maintaining this environment also raises costs, the benefit of planned data redundancy is acceptable distributed application performance.

IBM provides a controlled redundancy solution through its DataPropagator products. DataPropagator Relational (DPROP/R) will currently propagate from DB2 to DB2 databases running on OS/2, AIX, and OS/400. According to IBM, propagation to DB2 for HP-UX and Sun Solaris will be available in mid-1995. At this time, DPROP does not support propagation from the other platforms to DB2 on MVS.

One of the major components of DPROP/R is Capture/MVS (see Figure 2). Capture/MVS is a separate address space monitoring the blocks flowing to the DB2 log through the DB2 Instrumentation Facility Interface (IFI). It usually captures these log records directly from the log buffer area, therefore avoiding additional I/O to the log data sets. From this raw log data, Capture/MVS reconstructs the logical log records that detail the how data has changed. It refers to control tables to determine which tables have been registered for capturing. Each registered table has a corresponding *changed data table* into which Capture/MVS stores the table's changes.

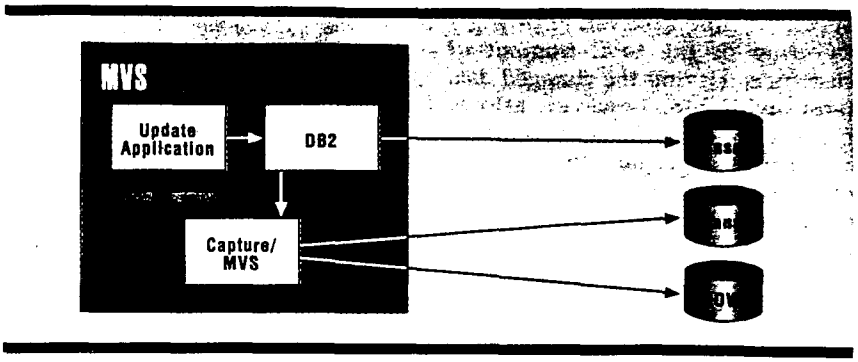


FIGURE 2. *IBM data capture process.*

Capture/MVS will, at an adjustable interval, commit all of the changed data tables with which it is working. Because the changed data tables are like a mass dump of all activity passing through the log—committed and uncommitted activity—Capture/MVS also maintains a UOW table into which it stores information about each transaction (such as time of commit, user ID, transaction ID, and so forth). Capture programs for OS/2 and AIX are due in April 1995, which will allow these platforms to be a source for propagation.

A second major component is the Apply Program (AP), shown in Figure 3. Once a user has subscribed to a registered table copy, the AP must satisfy the subscription. The AP can run at the primary or any of the remote sites. At the primary site, the AP would create what is known as the *consistent changed data table* by using the changed data and UOW tables created by Capture/MVS and appending four control columns to each row to provide different time slices of the consistent data. The consistent changed data table also opens up a host of opportunities for sophisticated data replication, which we will explore in a moment.

The AP can operate in refresh or update mode. Refresh, as the name implies, takes a complete source data copy and copies it to one or more targets. Update mode captures and copies changes only. As you can imagine, refresh mode is most appropriate for small or infrequently copied tables. Update is most often used for large tables with low volatility or those requiring close synchronization between the source and the target.

When the AP runs at a remote site, it can fulfill subscriptions by accessing data in the base table, the changed data table, or the consistent changed data table. In addition, any combination of these sources may be used to satisfy a request. The AP actually goes through an "optimization" step to consider which sources are available and which ones should be used to fulfill a request.

For instance, if a user has requested a point-in-time refresh, it may be satisfied most efficiently by accessing the base table and making a complete copy. The AP could then maintain that copy at a user-specified interval by accessing the changed data table (with references to the UOW table) to send only the updates that have occurred since the last apply.

The AP also uses these three sources to satisfy sophisticated subscriptions for aggregate data. Users can request two kinds of aggregates: base or change. For example, a base aggregate might SUM the number of customers in a table at a

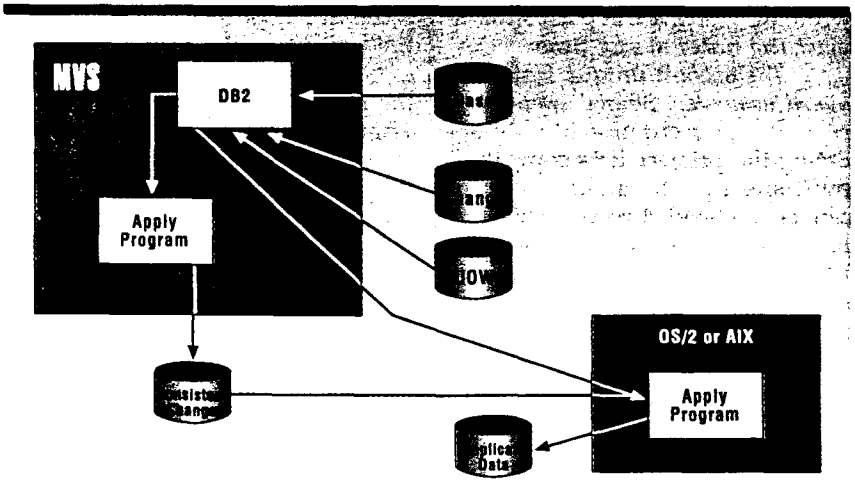


FIGURE 3. *IBM Apply Process.*

specified time interval, allowing a trending of the total number of customers. Having the changed data table available, however, allows more detail in the aggregation. You could aggregate the number of deletes and inserts to the customer table to get a better picture of your customer base's volatility rather than receive a simple absolute number. You could also answer such questions as "How many new customers were female?" or "How many of the customers we lost had a bad credit rating?"

The registration and subscription processes are managed by the final component: DataPropagator/2. This OS/2-based product is compliant with IBM's DataHub architecture. It provides a GUI interface for an administrator to manage the copy-management process. It is also the means by which subscribers, once authorized by the administrator, can request data copies.

ORACLE7 REPLICATION

Oracle was an early provider of replication through its Snapshot feature, which allowed a remote site to request a database copy from the primary site at a given point in time. This early implementation suffered from not being able to support a "changes-only" snapshot.

The snapshot capability has been extended with Symmetric Replication. You can define snapshots to contain a full master table copy or a defined subset of master table rows satisfying value-based selection criteria. Any changes to the master table since the last refresh are now propagated and applied to the snapshot at time-based intervals or on demand.

New in Symmetric Replication is what Oracle calls "primary site" replication. As the name implies, this facility propagates updates from a primary site to remote database copies. Like Sybase, Oracle follows a loose consistency model and provides the updates to the remote sites asynchronously. Unlike Sybase, Oracle does not scan the database log to detect updates; instead, Oracle depends on triggers and asynchronous stored procedures for propagation.

A trigger at the primary site fires on an update, insert, or delete. The trigger initiates the execution of an asynchronous remote procedure call (RPC) by submitting the request to a propagation queue within its local system, which, in turn, forwards it to a remote system or multiple remote systems for execution within a separate transaction. Asynchronous RPC transactions are executed on each remote system in the same order as they were committed to the local propagation queue. This asynchronous RPC mechanism is an added feature to standard Oracle7 PL/SQL stored procedures, which are synchronous and could still be used to perform updates via 2PC.

Sybase has chosen the loose consistency model

Oracle provides two new and, as of yet, unique features in Symmetric Replication (so new in fact that they will not be generally available until the first quarter of 1995). The first is *dynamic ownership*. With this feature, the "primary site" that owns the data moves from site to site, while ensuring that, at any given point in time, only one site may update the data. This feature provides a workflow type of capability: a piece of work (such as an order) flows from order entry, approval, shipping, and billing to accounts receivable. Dynamic ownership lets the responsibility for updating the order flow from department to department, while allowing all departments to view the order at any time.

The second unique feature is *shared ownership*. Shared ownership removes the restriction of only updating the primary data copy. With multisite update, all data copies can be updated directly. This feature is absent in both Sybase and IBM, which route remote updates back to the primary site to take place there.

Update conflicts are an immediate concern in such a scheme. Two sites are now capable of updating the same row at the same time. Work committed at one site but not yet propagated to a second site will conflict with an update done at the second site (to what is now the "old" data). Oracle's conflict detection routines work by propagating an update's before and after images. If a propagated update's before image doesn't match the row found at a remote site, a conflict is detected. Conflict detection is doable, but conflict resolution is more problematic.

Oracle will automatically invoke an application-specific (that is, user-written) conflict resolution routine to restore the replicated data to a "consistent state." This scenario may or may not be possible. For example, once an update has been committed, other work may be performed based on that result. It may be very difficult to come back and "restore" the replicated data. Oracle supplies some example resolution routines that work by timestamp or by adding or multiplying numeric updates.

PHILOSOPHICAL DIFFERENCES

A basic difference exists between the Sybase and IBM approaches to data replication. With the Replication Server, Sybase has taken the approach that all data must be replicated as soon as possible. The LTM captures data as it is logged and immediately sends it to all replicated sites. (If a site is not available, it stores the data on disk to be sent when communications are reestablished.) The reasoning behind this approach is that all applications at all sites must have the most current data available to perform their processing. The model is one of a central site pushing updates out as they happen, with the receiving sites required to handle the updates as they arrive.

This replication model makes the most sense in a transaction environment—and Sybase is aiming for this market. Sybase is driven to this approach, in my opinion, because many client/server implementations suffer from performance problems unless the server data can be located close to the client. Sybase's answer is to replicate updates so that many data copies can be available at many locations on the network for better performance.

IBM's approach is quite different. Propagation is intended to fulfill parts of IBM's Information Warehouse strategy. As such, it is

geared much more toward decision-support applications. In this model, the replicated site pulls updates from the primary site. The AP at the replicated site is under complete control of which updates come down, and when. This approach lends itself well to decision support, where it may be desirable to control a particular table's contents. For instance, users involved in producing a series of reports on sales history data would hardly be happy if the data's contents were continually changing.

Oracle provides some capability in both worlds: The snapshot provides a pull mechanism for decision-support environments; the asynchronous stored procedures will push the updates out in a transaction environment.

HIGHLIGHTS AND LOWLIGHTS

As you can imagine, each of these three approaches has positives and negatives. Which one is "better" for any given installation is likely to depend more on the application's requirements and other external factors than anything else. For instance, the logical choice may be dictated by what other pieces of architecture are already in place. IBM captures and propagates only to databases participating in its Distributed Relational Database Architecture (DRDA). If you have needs outside this environment, they will be hard to satisfy with IBM's solution. Likewise, while it is technically possible to write LTMs for other data stores, unless you have Sybase SQL Server in your shop, the Replication Server approach would be difficult at best.

Nevertheless, in this section we will cover several areas of strengths and weaknesses in these approaches. If you are contemplating which DBMS strategy to follow outside the mainframe, you should definitely consider the replication and propagation facilities. If nothing else, maybe some of the following ideas will give you something to bug (or praise!) your current vendor about.

☐ *Openness.* All products clearly state that their architectures are "open" and that operation within a heterogeneous world is possible. In reality, however, any time you cross the vendor boundary lines, things are going to get much more complicated.

IBM provides a controlled redundancy solution

In Sybase's case, any data store providing certain basic functions can serve as a primary store. All you have to do is write an LTM for it! Your customized LTM will capture updates off the log and put them in a format acceptable to the Replication Server. They can then flow down to a remote site and be applied as any other update, which is possible—but it is certainly not a trivial task. Replicated copies can exist outside SQL Server as well. The Replication Server can be "customized" to apply updates to any data store at the remote site, which means that the user can write the routines to perform the updates and the Replication Server will call them when needed.

For IBM, openness on the data server side is provided by the consistent changed data table format. A user can provide any data they wish as a consistent changed data table; it is then eligible to be used by the AP for propagation. IBM provides a companion product, DataPropagator Non-Relational (DPROP/NR), which will extract data from other data sources (such as IMS and VSAM) and format it to be loaded into a consistent changed data table.

DPROP/NR will also capture subsequent updates from the IMS log and put them into a consistent changed data table. A capture program for VSAM is planned for 1995. At the remote site, the AP works with data stores that are part of the DRDA architecture. IBM plans to add Sybase and Oracle as remote data stores through its not-yet officially announced DataJoiner product.

With Oracle Transparent Gateway, it is possible to replicate data from the Oracle Server to non-Oracle systems by defining tables in the non-Oracle system as read-only snapshots. Likewise, data in non-Oracle systems can be defined as snapshot masters for read-only snapshots in Oracle. Oracle also has plans for a variety of implementation techniques for Symmetric Replication to heterogeneous systems, but details are not available at this time.

☐ Plaguing any replication system is the problem of what to do with updates to the propagated data generated at the remote site. The only way to maintain complete data integrity is to rule out any local updating of propagated data; that is, all updates must be completed at the primary site.

Oracle's shared ownership feature deviates from this principle and cannot guarantee data integrity. Once an update has been committed, it may not be possible to "roll back" the update and resolve any conflicts, especially if subsequent transactions make decisions based on the committed data. Oracle's solution is to invoke user-supplied resolution routines that ultimately put the responsibility for handling conflicts back on the user.

Less need for updating exists in a decision-support environment such as IBM's. Typically, it is desirable to route the updates back to the primary site so that the changes can be pulled in a controlled manner.

Sybase is oriented more toward transaction processing; therefore, it usually assumes that transactions update data. Sybase provides two mechanisms to update the primary data remotely. One is through a remote stored procedure, which is executed synchronously with the original transaction. This synchronous execution introduces many of the problems that data replication was meant to solve (such as 2PC). The other option is using Asynchronous Transaction Propagation, where the remote procedure is executed asynchronously but requires that any failure be corrected manually.

☐ Potentially, IBM's pull model requires more resources at the primary site, especially as the number of replicated sites increases. When an AP runs remotely, it must access something—either the base, change, or consistent changed table at the primary site—to obtain the information it needs to update its data copy. In other words, 10

remote sites require 10 data accesses at the primary site (which is true with Oracle Snapshots as well). With Sybase's push model, the data is captured at the primary site once and distributed to *n* remote sites at that time.

☐ Hot spots, or heavy updating on a group of database rows, are a concern to DBAs if they often occur on the primary database, much less the replicated copies. The push technologies of Sybase and Oracle necessitate that each update done at the primary site be immediately sent to all replicates. Network traffic will quickly become a concern. Add in the potential for different sites' processing power inequalities (powerful primary and slower replicated sites), and this traffic could be the downfall of even the most well-intentioned replication system. IBM's pull implementation has an advantage because it allows you to wait until the smoke clears from the heavy updating before pulling down the results.

☐ Sybase and Oracle have an advantage for sites needing the most timely information possible at the replicated site (short of a tight consistency model). Even if IBM's AP were scheduled to pull continuously (the smallest interval is one minute), the capture program inserts updates into the changed data table selected by the AP. The Sybase offering captures the update from the log, passes it to the local Replication Server, which sends it to each remote site immediately. With Oracle, the asynchronous stored procedure is triggered as soon as the originating transaction commits.

☐ The data aggregation features of IBM's DPROP/R are much more powerful than those available in either the Replication Server or Symmetric Replication. The DPROP/R product allows SQL column functions as well as predicate logic (anything in a WHERE clause) to be applied to the changed data at the primary site before going over the network. Sybase provides only for predicate logic. Any data aggregation, if desired, is done at the replicated site.

☐ Sybase provides multiple primary sites for a single table. If a table is logically partitioned, each replicated site could contain the

Oracle was an early provider of replication via its Snapshot feature

entire table, but could be the primary update site for one of the table's partitions. For example, each region of a company could be responsible for maintaining the customer data for customers in its region. Yet all regions would have a copy of the entire customer table through the Replication Server.

Oracle takes this approach a step further with its shared ownership model, which allows any of the sites to update any of the data. Technically, this approach would be possible with DPROP/R if all participants were mainframe DB2 sites. With no capture program currently available for the DB2 products on OS/2 or AIX, the more homogeneous environment of DB2 propagation to other platforms does not provide for multiple primary sites.

☐ With IBM capturing and storing every change to primary data in the changed data table, the user has the added functionality of being able to use it for auditing purposes. Such a side benefit is not possible with the Sybase implementation. Oracle's changed data capability is unclear at this time.

☐ IBM and Sybase compensate for integrity problems in similar ways. Bowing to the fact that replicated data may be updated locally outside the replication system's control, IBM will automatically correct certain problems on the fly. If an update is sent down for a row that doesn't exist, it will be turned into an insert. Likewise, inserts can be turned into updates. Sybase provides the same capability through its Automatic Correction option. If you do not choose this option in Sybase and an integrity problem is found, an error is logged and user intervention is required. With Oracle, remote site update capability is a feature that comes with the data integrity concerns mentioned earlier.

☐ IBM, by taking advantage of DRDA, uses limited block prefetch between the primary and remote sites. As the AP starts to fetch a set of data, the primary site blocks the rows to reduce the amount of communications protocol required. With Sybase and Oracle (excluding snapshots), each row is captured and sent individually from the primary to remote sites.

WHAT'S RIGHT FOR YOU?

The problems inherent in full distributed data processing makes it likely that replication technology is here to stay. Sybase, IBM, and Oracle have capable offerings that will surely be enhanced through subsequent releases.

Sybase is oriented more toward a transaction-processing environment and, therefore, has the edge in those situations. Its push technology is the fastest way to propagate data to remote sites, with changes often reflected within seconds. Propagation is completed with the least amount of resource demand on the primary site.

IBM far and away has the better product in a decision-support environment. The capabilities realized by using the pull technology against one of three sources at the primary site are superior to Sybase's straight capture and push technology. It offers sophisticated data aggregation features that can reduce the amount of data flow over the network.

Oracle covers a bit of both environments by offering the push technology with primary site propagation and the pull technology with the snapshot capability. The advanced features of dynamic and shared ownership are not generally available at this time, and so their impact on the market is hard to judge.

Whatever your preference, do not overlook these offerings as you distribute data to various platforms. As the number of remote locations increases, you will quickly learn that distribution is no longer as easy as cutting a tape here or setting up a file transfer there. Why not let these and other products help you stay out of the Information Junkyard? IIII

Doug Stacey is director of data access and connectivity products for Comdisco Inc., based in Rosemont, Illinois.