George's Quarterly

Fall 1994

Inside this issue...

Published by:

DCI

DCI, 204 Andover Street
Andover, MA 01810 USA
(508) 470-3870
FAX (508) 470-1992

Dear DCI Friends and Colleagues,

Fall greetings! This issue is a bit late in arriving due to a tremendously busy schedule. DCI has been running full-steam ahead in designing and coordinating a multitude of new conferences, expositions, and seminars. Many of our newest offerings are described in an article starting on page fourteen. The largest, to name a just few, include *WEB WORLD: Corporate Solutions Today*, *Networked Multimedia EXPO*, and *PC Card '95 Conference and Exhibition.*

Starting on page two is the second part of an article we began in the Summer issue of *GQ, **Replication**: The Next Generation of Distributed Database Technology*. In this issue, topics covered include: TP-R replications, peer to peer architectures, TP-R and fault tolerance, transparency and richness of function, replication timing, database configuration, and a summary of replication benefits.

As always, keep in touch as I do enjoy hearing new stories. I am reachable through DCI voice mail (508) 470-3870, ext. 403 or CompuServe 74407,2472.

Talk to you next year!

Geo Schussel

This article is the second and last installment in a series on replication.

TP-R Replication: Peer to Peer & Master/Slave Approaches

Although many DBMS vendors are talking about replication offerings, it would be a mistake to assume that replication is a commodity. Different architectural approaches to the implementation of replication provide fundamentally different capabilities. Not only are there important replication server differences between DSS-R and TP-R approaches, but within each of these architectures there are important differences.

TP-R approaches have been implemented with two fundamentally different architectures by ASK/INGRES and Sybase. ASK/INGRES has built its replicator on a peer to peer architecture approach. Sybase uses a master/slave approach.

TP-R replication is primarily concerned with creating a single image of a database across distributed autonomous sites and preserving database integrity in near real-time processing. The overall integrity of databases is preserved by forwarding data changes resulting from single-user transactions.

Master/slave vs. peer to peer

All data replication, regardless of vendor, copies data from sources to targets. Master/slave approaches replicate data from master to slave, requiring updates to successfully complete at the master before the transaction is considered a success (as far as the application goes). On the other hand, updates in peer to peer approaches can be made to any data location and then copied into other locations. A transaction is successfully completed as soon as any one or combination of locations is able to update one complete copy of the affected data. Peer to peer allows all locations to own and manipulate any data, broadcasting changes as required.

In the master/slave architecture, every table or table fragment is assigned to a primary site. If the primary table's database server fails or if access to that server from the network (where a transaction updating that table has occurred) is denied, replication doesn't occur and the transaction is queued. This can present a problem for remotely generated transactions because those processes cannot update local or other sites, until they are first routed synchronously through their primary tables.

Master/slave approach to TP-R

The master/slave approach to TP-R has the following characteristics:

- It's simpler for a vendor to implement (from the replication server point of view) because it eliminates the potential problem of update collisions (explained next page).
- Because its implementation is simpler and more straightforward than peer to peer, in some circumstances applications will run faster because of lower DBMS overhead.
- It introduces a single point of failure that can lower the overall system

availability as compared with the peer to peer approach.

- It's a less general solution than peer to peer.

Although the Sybase architecture is master/slave, the vendor states that its Replication Server can be set up to support a peer to peer approach. As is discussed below, collision detection and resolution software should be provided by any system that supports peer to peer transaction replication.

Sybase normally requires that updates to slave databases first be routed through the master database. This eliminates the need for collision detection and resolution. However, if you want to build a peer to peer architecture with Sybase technology you'll have to write your own:

1. collision identification software,
2. collision resolution logic, and
3. logging transfer manager (including recovery).

Doing this would be work well beyond the capabilities of the typical data processing shop.

Peer to peer and TP-R

The peer to peer architecture, of which INGRES is the only vendor at this point in time, is the most general and powerful approach to TP-R replication. It is closest in capability to a true distributed DBMS in that there is no limitation on where data can be located or updated. And yet, because we're talking about a replication server which uses many individual two-phase commits to broadcast data changes that are asynchronously distributed from the originating application, peer to peer is more fault tolerant than a distributed DBMS.

A problem that is related to using a peer to peer replication approach, however, is the possibility of "collisions." Collisions occur when two different originating nodes update two different physical copies of the same logical data with two different transactions. When the replication server attempts to broadcast changes from each of those originating sites, it will become aware of this conflict in updates and need to begin a process of reconciling the differences.

Collisions with a Peer to Peer Architecture

A collision is when the same record, which is physically replicated at two or multiple sites, is updated during the asynchronous latency period. In other words, after the time a first update has happened, a second update occurs which is processed at one site before the propagation of the first update has been completed. So although a peer to peer approach provides the most general solution for transaction distribution, it requires software for collision resolution.

When a collision occurs there is no way to construct an application-independent approach that can recover all different types of databases. However, the replication server can and should have collision resolution logic. First and most important, collision resolution requires that the system provide notification that a collision has occurred.

From the moment any transaction is committed, the replication server has to keep track of all of the processes that happen further in the processing and distribution of that transaction. That's because in the event of a collision, this in-

formation has to be available to properly resolve the collision.

The replication server should support multiple options for the database administrator (d.b.a.) to choose from in resolving the conflict. Examples of resolution possibilities include:

1. The initial update has priority. Roll-back the conflicting (and later) transaction with necessary messages to designated parties.
2. The last update has priority. Overwrite the conflict and send the necessary notices.
3. Resolve the conflict by firing a user-specified trigger.
4. Halt the replication process and send a message to the d.b.a.

In order for a number of these processes to work, it's helpful if there is a distributed time service available because current replication servers don't provide this. The replication server depends on the separate operating system clocks. If they aren't synchronized, errors will result. An important new facility for this service is OSF's Distributed Computing Environment (DCE) which provides the necessary synchronization.

Experience to date with users of peer to peer replication indicates that if the replication timing chosen is ASAP, and if your databases have been properly designed for replication, the volume of collisions is likely to be very low. Those conflicts that do occur can be handled by rules in a collision resolution software module with log entries for manual review, or by manual review. Future capabilities for replication servers in this area may include expert systems to help resolve collisions.

Collisions don't happen with a master/slave architecture such as Sybase's. This is because the transaction is simply not accepted unless it can be committed at the master site, or what Sybase calls a "clearing house."

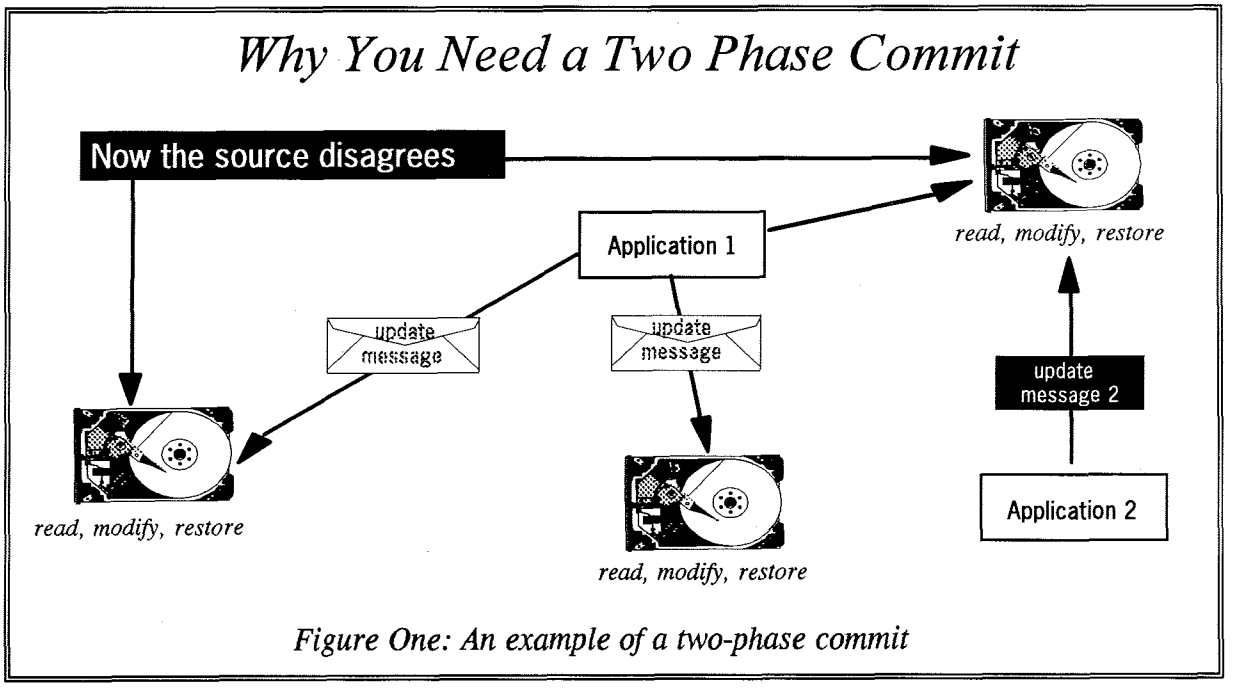

Figure One: An example of a two-phase commit

It might be useful to refer to Figure One (*previous page*) to re-analyze what would have happened had peer to peer replication been used. In that case, the application, would have been accepted and considered successful at the completion of its first database update. That's a powerful performance advantage. Later, however, further processing on the network resulted in a collision. Some further processing and/or manual involvement, then, will be required to recover the multiple database copies in a consistent way.

TP-R and Fault Tolerance

One of the principal benefits of all replication approaches is added fault tolerance for a distributed computing environment. Fault tolerance provides the overall system with a capability of continuing to function when a piece of the environment is down.

When something breaks, then, the system working in combination with the d.b.a. should provide as much assistance as possible in the recovery process. (Mike Stonebraker has used the phrase "failover reconstruction" to describe when this recovery process occurs automatically under software control). Necessary steps in the failover reconstruction process should include:

1. understanding what is broken,
2. understanding what or how the break occurred,
3. determining how to fix the damage and reinstate the broken pieces,
4. bringing the broken pieces back on-line,
5. making sure that the recovery of the database(s) results in consistent data in those database(s).

The highest level of fault tolerance will be from a system supporting peer to peer replication. That's because the system considers an update to be successfully completed when it has completed a database update at *any* peer site. The site that is updated is like a floating master in this case. The replication server will queue the updates to all other data locations.

...Collisions don't happen with a master/slave architecture such as Sybase's...the transaction is simply not accepted unless it can be committed at the master site, or what Sybase calls a "clearing house."...

In a master/slave architecture, if access to the master is denied, then the update is not allowed from the application. When the master location becomes available, it then becomes updated. After the master has been updated and if there is failure elsewhere, the replication server queues the updates to the slaves until they are available. This system works as well as a peer to peer approach unless it's the master node or network that fails.

In either case, it's important that your system provide the necessary utilities to allow the rebuilding of remote databases from information on the local log as well as database information on other remote databases. One key utility should be able to "difference" replicates —in other words to look at a master and slave or two peers and determine if inconsistencies exist.

Transparency & Richness of Function

For a replication server product to be successful, it has to provide enough added function over what customers have developed and should provide that function transparently to customers. There is a significant difference in the amount of replication function provided by various DBMS vendors and in the ease of implementing replication and its various features.

Some products require significant programming with database triggers or database calls to implement replication. Most of the current replication functionality in Oracle 7, and much of the service available through Sybase System 10 Replication Server, requires programming with RPCs or DBLib calls by the distributed data base administrator (d.d.b.a.). Setting up database replication with INGRES is easier in that a there is a configuration manager that offers a three-step forms-based approach to defining the replicated environment.

The TP-R Schema

In order to provide transparent replication services to applications, the d.d.b.a. needs to be very much aware of the use of a replication server and needs to have designed the database in a manner that is conducive to distributed operation. In practice this means that denormalized and/or aggregated data should *not* be replicated in TP-R situations. Such derived/aggregated data should be computed at each site from the basic data contained in a transaction.

To see this point more clearly, let's discuss the banking example below (*Table One*). It illustrates a process that spans three periods of time (A, B, C) and three branches of a bank (1, 2, 3).

We're looking at one customer's balances after withdrawals are made during a period of time when the network to one replicated site is down.

- At time A, the network is entirely up and the customer's balance (100) and current transaction (none) are identical at all three bank sites.
- At time B, the network link to bank 1 is broken. The customer makes a withdrawal at bank 2. That transaction is replicated into Bank 3 and the balance from 2 is also replicated into 3. Bank 1 still has the old information since access to the updates is unavailable.
- At time C, the customer makes a withdrawal at Bank 1.

	Bank 1			**Bank 2**		**Bank 3**	
Time	*Balance*	*Transaction*		*Balance*	*Transaction*	*Balance*	*Transaction*
A	100			100		100	
B	100		X	60	-40	60	-40
C	70	-30	X	60	-40	60	-40

Table One

At any time after this withdrawal, an attempt to reconcile the balances among the three banks is going to fail. That's because the account balance field in this example is aggregated (and de-normalized). Replicating balance information is going to cause integrity problems with the data bases.

Let me repeat, then, an important rule in the TP-R environment—do *not* replicate aggregated or denormalized data. In our banking example, if the system had simply replicated the transaction amounts, normalized data, each site would be able to recover correctly from a collision like the one illustrated by using a time-order to sequence and process (and then compute the balances). In general, a good rule for distributed processing is to use local database triggers to handle computed amounts such as account balances.

Replication Timing

Your application shouldn't need to worry about the timing of the asynchronous distribution of data to target sites. Getting this functionality from your replication server also shouldn't require you to do programming.

The replication server, be it TP-R or DSS-R, should also provide several alternatives for timing. Examples are:

1. immediately, or as soon as possible (ASAP). In this case the data is moved through the queues and replication server as fast as possible.
2. scheduled, as determined by the system administrator. In this case, data remains in the replication server until it is scheduled for distribution.
3. triggered, by user defined criteria such as an event happening, the number of records exceeding a limit, or time of day. When that trigger is fired, the server moves the data to the distribution queue for remote processing.
4. under manual control.

l dictate the type of timing used in replication. For operational systems that expect to be updated with near real-time transactions, the best approach is likely to be ASAP. There is no additional processing overhead attached to ASAP replication in this case because the user is likely to be in a situation where the copy distribution is under two-phase control for each updated site (to preserve transaction integrity). In such a case, then, there is no processing savings attached to batching the transactions (although transmission at night might offer savings).

For decision support or period accounting types of systems, a stable database that is consistent throughout may be preferable to having the most current status. In this case, for reasons discussed above, scheduled replication may be preferable.

Database Configuration & d.b.a. Utilities

Managing a distributed database is significantly more complicated than running against a monolithic single location database. The d.d.b.a. has all of the design and implementation issues of a single location *in addition to* the added complexity of distribution, network latency, time shifts, and remote administration.

The d.d.b.a. is a new job function in addition to local d.b.a.'s. The following are examples of work the d.d.b.a. will perform:

- Designing and planning the replication system, including how and when data is shared amongst users. It's only after this work has been done that the local d.b.a. can input the necessary information to set up the replication system.
- Coordinating the installation and system configuration among its various sites.
- Monitoring the operation, and performance and recovery of the system from an enterprise, rather than a local, perspective.

Some ideas to remember as you consider implementing a replicated database environment are:

- Set up a plan and understand the rules for distribution of data before the implementation begins. Implementing replicated databases is not technology amenable to "let's try it and push it around a bit" approaches. It's necessary to have a good plan in hand before you begin or you will get lost in the middle of building the replicated environment. If your plan is good, the implementation can proceed in incremental fashion, however.
- Make sure that your d.d.b.a. has good forms-based or graphical utilities to assist in the database configuration and in the management of the ongoing network. For example, INGRES comes with forms-based management utilities. IBM and Sybase have GUI-based management utilities. These facilities should be able to manage all aspects of a replication environment from a single desktop that's moveable and can be anywhere on the network. Some points to carefully consider:

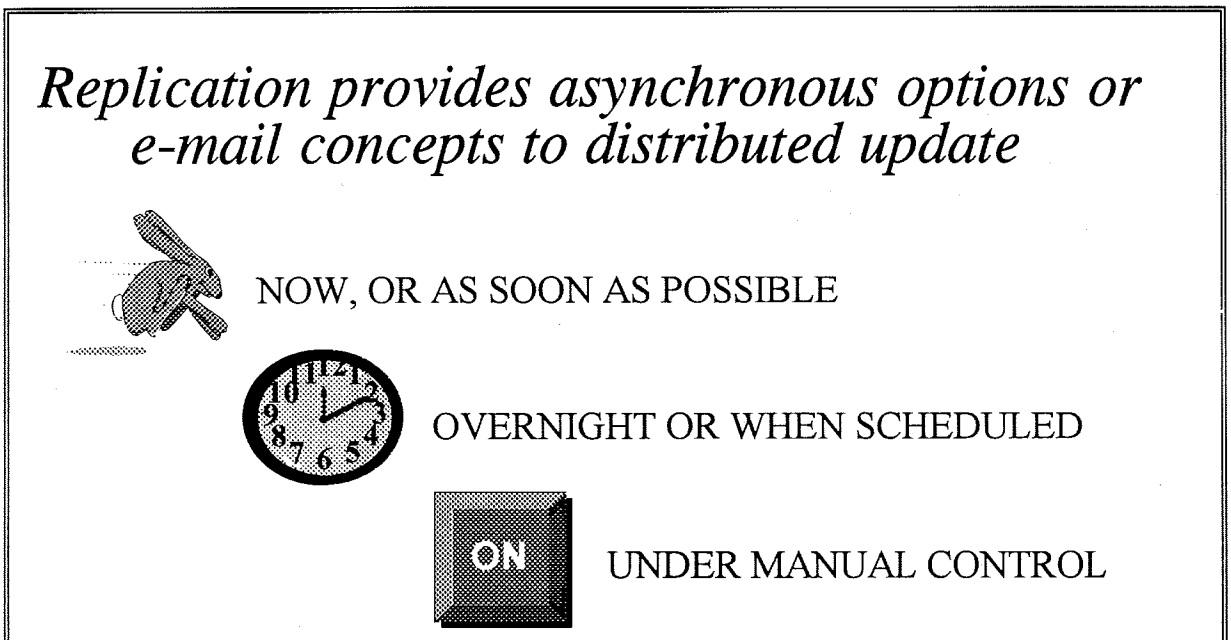

Figure Two: Replication offers various timing options

1. How do you specify enhancements to the data? Do you have to learn a new language for this function?
2. How is the replication setup handled? How much automated support is provided to the d.d.b.a.?
3. What is the support provided for failure management? How much recovery is automatically handled and how much d.b.a. intervention is required?

- Your utilities should be able to answer questions like:
  1. What tables are at what nodes?
  2. What columns are at what locations?
  3. What rows are at what locations?
  4. Where are transactions routed to?
- You should be able to change the database configuration on the fly without bringing the database or replication operation to a standstill.
- There should be a mail-based error notification system. This allows management of the distributed enterprise from any node on the network.

...The essence (and the bane) of distributed database is the two-phase commit. What the two-phase commit accomplishes is a synchronized locking of all pieces of a transaction....

Replication into Heterogeneous DBMS

Today, there are no standards that apply to replication across diverse products. And there are no standards bodies working on this issue. As a result, issues such as utilities and recovery are handled quite differently in different vendor's products.

All of the major DBMS vendors are moving toward opening up their replication capabilities to foreign DBMS. Digital, Oracle, Sybase, and IBM are focusing their attention on links to each other and other relational DBMS products. IBM, INGRES, and Sybase have published their two-phase commit protocols which allows their users to participate in heterogeneous distributed database approaches with products from other vendors.

Both Sybase and INGRES have links *to* non-relational DBMS in their target replication capability. Normally if the vendor supports a gateway to that DBMS, then it can serve as a target for replication. That includes IMS, RMS, VSAM and other environments for both of these vendors. The gateways to non-relational DBMS don't require special coding (such as RPCs) and are valuable in allowing the integration of new distributed systems with older applications.

As a general rule, replication *from* a foreign DBMS into a replication environment such as INGRES or Sybase is only available now if the user is willing to program that functionality. One important exception is an IBM offering which allows replication from IMS into the DB2/DRDA world.

Anyone contemplating the acquisition of replication technology should understand how your vendor will assist in

migrating to a heterogeneous DBMS environment. Almost no organization today uses one DBMS exclusively. Heterogeneity in database and file management approaches is likely to increase in the future. Gateway solutions, of course, are not the same as a replication and two-phase commit processes that transparently operate over multiple DBMS. The real world is multi-vendor, multi-department, and multi-network. Replication technology that can operate well across heterogeneous DBMS is something that DBMS users will want.

Summary of Replication Benefits

Better Response Time from Local Data

A replication server can be instrumental in allowing more efficient usage of a company's computers and network. By shifting data to the local site where it's needed, companies can insure that important applications are available at all times. The response time achievable from local data access can be significantly improved over response that depends on access from a distance. In addition, replication is more fault tolerant than distributed DBMS. That fault tolerance results in more consistent processing of transactions with the result that the overall database is up and responsive more than the equivalent configuration would provide if it were a distributed DBMS.

Replication for Hot Standby Backup

Replication can provide the architecture for backup that can enhance your system reliability in a local (and/or WAN) environment. Replication, enhanced with hot-standby software, operates by monitoring the performance health of your primary server, while transactions are backed up on the replication server. When there's a failure on the primary processor, the backup is

Replication can provide you with remote backup capability

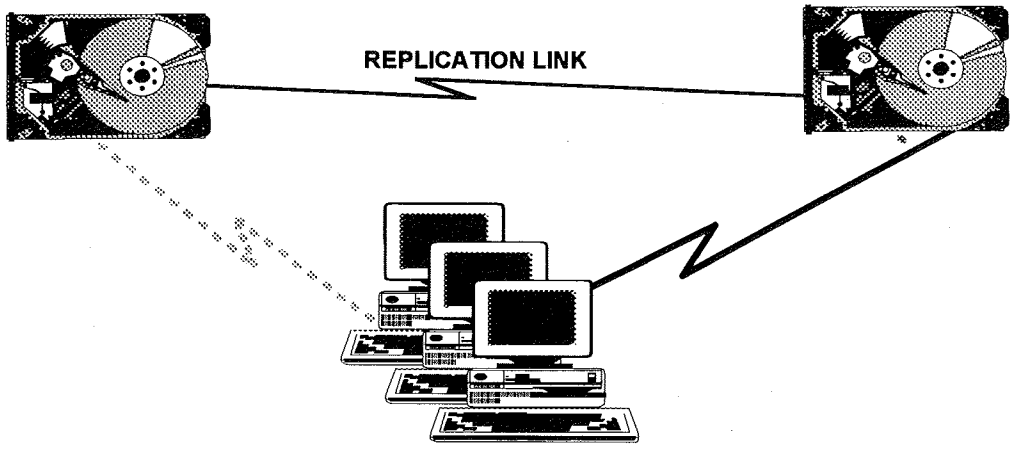

Figure Three: Replication can provide backup to enhance your system reliability

immediately available. The system automatically switches to the backup and designates another machine as the new backup replicate.

Data Availability Such as Separate Servers for Separate Functions

Individual workgroups can now have their own replicated databases. This means not ever having to say you're "sorry" for network propagation delays. Replication can enhance performance and provide load balancing locally or over a WAN. As an example of this, two replicate servers could allow queries to be channeled to one machine while updates and production work are channeled to the other. The query server will have accurate information that is exactly current or somewhat dated, depending on the speed of replication chosen by the user. With DSS-R approaches, the database copies can be enhanced for decision support. Data can also be replicated from legacy applications and made available now to new styles of processing across the network.

Decision support types of applications are natural replication candidates, because if they're distributed, replication can greatly reduce WAN traffic.

Splitting the Workload for Capacity Relief

As companies migrate to decentralized operations, they naturally want their computing support to follow the same form. As the workload is distributed, it is split among multiple servers. There are significant cost savings attached to using multiple smaller machines to process work. Replication, done intelligently, can reduce network traffic and allow the user to derive benefit from what would otherwise be unused CPU cycles. Another way to look at this is that replication allows easy local data

Replication can reduce network traffic, provide better local response, and lessen host processing.

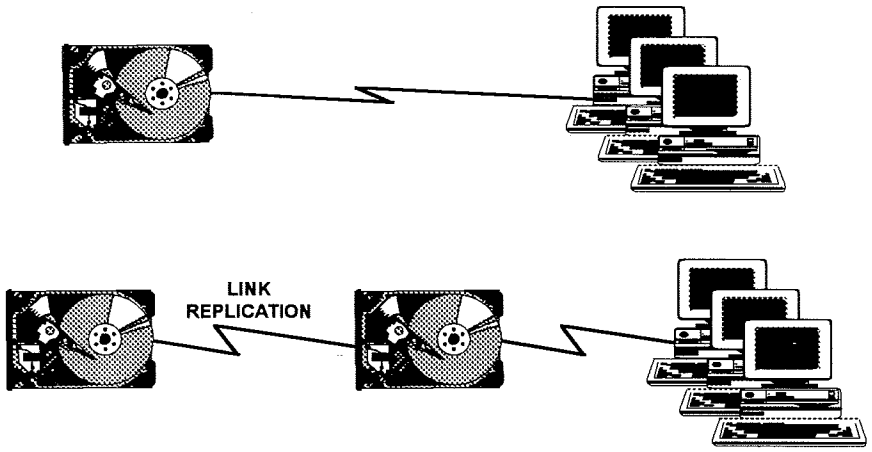

Figure Four

access at remote sites. This, then, allows:

1. a decrease in response times,
2. a reduction in wide area network traffic, and
3. the establishment of local autonomy which can take over in case of network or server failure. A key to achieving this advantage is to use a peer to peer type of replication service. This is so that when recovery occurs the completed local updates can be properly propagated to other locations of the same data.

Non-Stop Processing & System Fault Tolerance

Replication is an important technique for increasing the availability or uptime of network-based computing. Redundancy is the fundamental engineering approach for increasing reliability and replication can be used exactly for this purpose.

Imagine a retail operation where sales offices are widely distributed and inventory is kept at a few major warehouse locations. If the warehouse information is replicated at the sales offices, then it's possible for the sales office to accept tentative orders even if the network link to the local warehouse is broken. The sales office can accomplish all of the processing necessary for a sale except for a final confirmation without access to the central source inventory data.

This kind of capability provides for a

Replication can split the load across several machines providing more capacity and less network traffic.

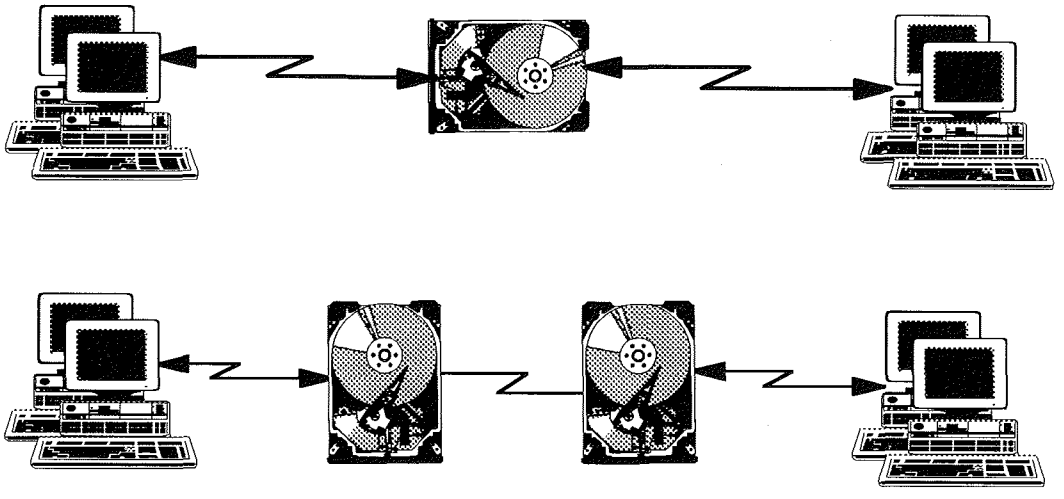

Figure Five

higher level of customer service than what could be provided by a system operating off a single central database with communication links to the distributed sales offices. For a distributed operation, then, replication of both TP-R and DSS-R types allows for higher system availability than a monolithic model.

Conclusion

It's a Complex Environment

The benefits of a properly implemented replication scheme can be very substantial. The complexity however, in both a managerial and technical sense, of a distributed environment is much greater than that of a local monolithic environment. This is especially true for TP-R environments. Data collisions may occur with peer to peer approaches; the recovery process that this implies requires the cooperation of excellent software and competent administration.

Your Database Administrator is a Key Resource

It's wise to invest the necessary resources to make sure that the combination of local and global d.b.a. resources is adequate for your environment. Your d.b.a. will have to create a data base design that is correct for replication and tested in the distributed environment. In an operational sense it's important to not shortchange the time it takes for your d.b.a. to become an expert in diagnosing and resolving problems in this environment. You should also seriously consider consultant assistance, probably from your DBMS vendor, as part of the first project.

Replication can support non-stop or 7 x 24 operation. Take database one off-line and optimize, revise indexing, install new apps, etc.

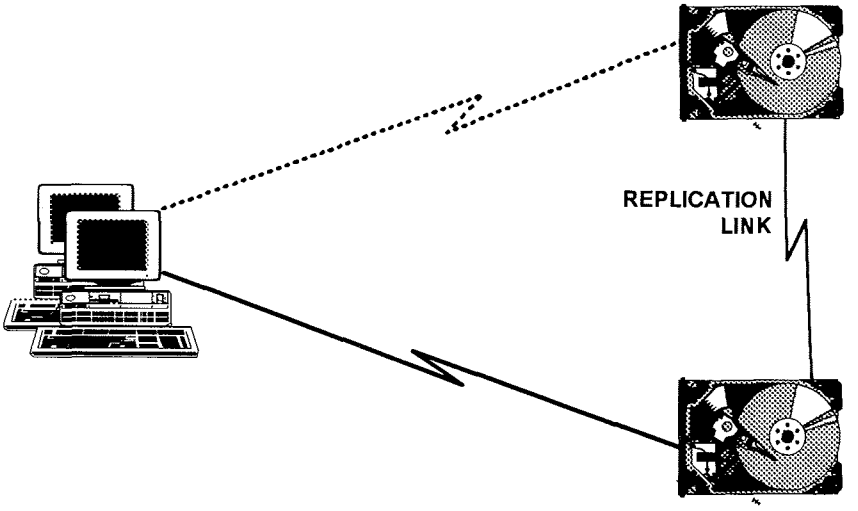

Figure Six

Your Approach Should be Cost and Benefit Based

Make sure that you understand the architectural, currency, data integrity, and performance implications of a DSS-R or TP-R based approaches. Different approaches from within any one vendor's product line and/or between vendors mean that different technologies have very different cost, performance and integrity results. You should have a DBMS that supports the different requirements of your application environment.

Managing distributed data through replication and copy approaches is non-trivial and will require competent technical management. Even evaluating the different currently available technologies will require an analyst of top caliber.

Because implementing distributed systems offers so many combinations of technology and benefit, you'll need to do some careful management analysis to understand how these approaches can support your business requirements. Those business benefits should be measured against the costs of the software and management necessary.

Keep it Simple, Especially at First

It's wise to begin implementing a distributed database with a single vendor. However, If you have a heterogeneous DBMS environment, be sure to understand how your vendor can support a multiple DBMS approach.

■

NEW DCI Shows and Seminars

WEB WORLD: Corporate Solutions Today

January 30—February 1, 1995,
Lake Buena Vista, FL

As the fastest growing facet of the Internet, this technology is something that no corporate entity can afford to ignore. The World Wide Web, enabled by NCSA Mosaic and similar browser technologies, allows information stored in a variety of formats to be accessed and processed by users of all levels.

DCI is pleased to announce this three-day conference to be chaired by Dr. Jay C. Weber, Director of Research and Development at EIT Corporation. The focus of the conference will be on how your firm can use the Web technology to gain a competitive advantage. You will learn how to find what is available, get a jump on how to build your own Web applications, and let people know you are on the Web.

There will be three different conference tracks which will focus on:

- Living in the Web: User Perspectives
- Building Webs: Web Access & Information Providers
- Weaving in Information Space: Developing Web Environments

High-powered sessions will include information on such topics as:

- What resources are currently available to corporate users?

- How can I use the Web to enhance my business?
- How do I navigate to find the information I need?
- There are currently two leading languages used on the Web, SGML and HTML, which one will pay off?
- How do I secure and protect my data?
- What are firewalls and how do they work?
- How to design safe, cost effective, and efficient payment infrastructures?

Either call DCI at (508) 470-3880 for more information, or check out the conference brochure on the Web at: http://www.oec.com/DCI/

Networked Multimedia EXPO

July 18-20, 1995, San Jose, CA

The cornerstone of electronic commerce is the use of networks and multimedia applications. The implementation and management of these emerging technologies are the responsibility of IS professionals. Yet, until now, these managers have very few ways, if any, to educate themselves on this advanced technology.

Networked Multimedia EXPO, created by DCI and Tansy Associates, steps up to the challenge of providing this desperately needed information.

PC Card '95 Conference and Exhibition

April 4-6, 1995, San Jose, CA

PC Card '95, the PCMCIA Conference and Exhibition, promises to be the best venue in the world for learning how PC Cards are simplifying our increasingly complex computing world. PC Card '95 features a three-day program designed to demonstrate PC Cards and how they can contribute to the effectiveness of your computing strategy. This comprehensive program contains specific sessions for all levels of interest from basic lessons to advanced tutorials.

PC Card '95, being held at the San Jose Convention Center next April, also gathers together the leading industry vendors in a two-day exhibition which will showcase the diversity of the industry.

Navigating the Client/Server Road Map

February 6-7, 1995, Boston, MA

For those who are new to client/server technology, this seminar explains how best to get started. For people who have already had some experience, you'll find the Client/Server Road Map to be extremely valuable in providing a framework for evaluating the multitude of client/server software offerings currently available.

At the end of this two-day seminar taught by Jeff Tash, you will know:

- how to evaluate the numerous client/server software products currently on the market.
- about the industry's hottest client/server application development tools including PowerBuilder, SQL Windows, Visual Basic, Uniface, Object View, Smalltalk, Enfin, Dynasty, Forte, and many more.
- a proven application development methodology that uses object-oriented techniques to create reusable software

modules that can be snapped together and pulled apart like *software legos*.

Using RAD on GUI Client/Server Projects

February 10, 1995, Boston, MA

At this one-day seminar, developers, project leaders, and technical managers will learn the fundamentals of successful process management. Seminar instructor Christine Comaford will also discuss how to address client/server application development using interactive user-centric approaches.

Attendees will learn Rapid Application Development principles within the framework of three paths—infrastructure, management, and development.

Building Enterprise-Class Applications with Client/Server Technology

April 25-27, 1995, Chicago, IL

This three-day seminar provides attendees with practical, objective guidance on how to make the transition to a client/server computing environment. It will guide you through the steps involved in moving from a centralized computer system with a proprietary architecture to a distributed, client/server system in an open systems environment.

Instructor Pieter Mimno will focus on the key enabling technologies that can be used to develop applications rapidly at low costs.

The Second Generation of Client/Server Computing

March 1-2, 1995, Washington, D.C.

The approach of this seminar is to examine the real-world problems that organizations are trying to solve for which there are no elegant, off-the-shelf solutions available. This two-day seminar gives pragmatic "how-to" coverage of client/server computing and distributed data.

Topics to be covered by Instructor Herb Edelstein include:

- Where the first generation of client/server computing fails
- The potential savings and hidden costs of client/server computing
- How the change in computer usage is fueling the move to client/server computing
- Why object-oriented tools are not a silver bullet for productivity
- When three-tier and two-tier client/server architectures are appropriate
- Distributed query processing
- Distributed transaction processing

***George's Quarterly*, Fall 1994, Volume 1 Issue 4**
Editor George Schussel, Managing Editor Stacey Griffin

George's Quarterly (*GQ*) is published by DCI in four annual editions: Winter, Spring, Summer, and Fall.