



Replication Strategies: Data Migration, Distribution and Synchronization

A Sybase® White Paper

available

Table of Contents

Abstract	1
Sybase Replication Server: A Practical Architecture for Distributing and Sharing Corporate Information	1
Introduction	2
What is Data Replication?	2
Early Data Replication: Dump and Reload	3
Synchronous Replication: Two-Phase Commit	3
Non-Transaction-Based Replication: Snapshots	3
Non-Transaction-Based Replication: Database Triggers	4
Transaction-Based Replication with Triggers or Rules	5
State-of-the-Art Replication: Sybase Replication Server	5
Where Does One Use Replication Server?	6
Example 1: One Primary - Multiple Secondary (Decision Support Replicates)	7
Example 2: Warm Standby Database	8
Example 3: Multiple Primaries - One Secondary (Corporate Data Consolidation)	9
Example 4: Peer-to-Peer Configuration With No Update Conflicts (Corporate Data Sharing)	10
Example 5: Peer-to-Peer Configuration With Update Conflicts (Corporate Data Integration)	11
Example 6: Upgrade an ASE Application	11
Example 7: Non-Sybase Data Replication	12
Example 8: Replicate all or part of the table	12
How Does Replication Server Work?.....	13
Transaction (Data Change) Detection - Replication Agents	13
Data Propagation and Routing Across LANs and WANs	14
Data Delivery to Sites with Replicate Data	15
Replication System Configuration and Management	15
What if a System or Connection is Unavailable: Stable Queues	16
Modifying Replicate Data: Distributed Updates	17
Why Sybase Replication Server?	19
Replication Server Benefits:	19
High Performance.....	19
High Performance Data Capture: No Triggers or Rules	20
High Performance Data Delivery	20
High Performance Data Access	20
Consistent Information Delivery.....	21
Easy Centralized Administration.....	21
High Availability of Data	21
Heterogeneous Integration	22
Local Autonomy	22
Summary	22
Appendix: How Does One Set Up a Replication Server Environment?	23
Example 1 Revisited: Decision Support Replicates	24
Example 2 Revisited: Corporate Data Consolidation	25
Example 3 Revisited: Corporate Data Sharing	26
Example 4 Revisited: Corporate Data Integration	27
Example 5 Revisited: Non-Sybase Data Replication.....	27

Abstract

This paper describes Replication Server®, designed specifically to meet the needs of enterprise wide, client/server computing. Replication Server allows you to share information across an enterprise by replicating it to and from different hardware platforms and data sources without losing the transactional integrity of the data. This is particularly important when designing and implementing large-scale data migrations. Replication Server's ability to maintain a constant, uninterrupted flow of data to and from the database uniquely qualifies this technology to manage migration, distribution and synchronization for enterprise-critical systems.

Replication Server provides a safe way to upgrade or migrate existing systems and platforms. By replicating data during migrations, customers can migrate existing database data without interrupting critical business applications. The applications can be switched to the new version or platform once the new database server is populated, applications are tested, and any new features are implemented.

Replication Server also supports replicating data to and from non-Sybase data servers. Data can be replicated to non-Sybase data servers such as Oracle, Informix, IBM DB2, and Microsoft SQL Server using Sybase DirectConnect™ gateways. Transactions can be captured and forwarded from non-Sybase data servers using Sybase Replication Agents™. Data can also be replicated from a non-Sybase source, through Replication Server to a non-Sybase destination.

Replication Server provides warm standby capability with a pair of databases, one as the active database and one as the standby database, is supported by Replication Server functionality. As clients update the active database, Replication Server copies transactions to the standby database, maintaining consistency between the two. Should the active database fail for any reason, you can switch to the standby database, making it the active database, and resume operations with little interruption.

Using a reliable store and forward technology, Replication Server accommodates the reality that architecture changes occur, and network connections and network components can fail. If a remote site is accessible, the information is forwarded to that site; if one or more remote sites become unavailable, the information is stored until the connections are re-established, at which point copies are automatically resynchronized. Corporate sites can continue operations using their local copy of the data even when remote sites are inaccessible.

Critical to making such a distributed environment easily manageable is the Replication Server Manager™, a powerful systems management tool with an easy to use graphical user interface. Available at no extra charge with Replication Server, this systems management tool allows administrators to manage and monitor from a single site all distributed components of the enterprise client/server replication environment.

This paper begins with an introduction to the concept of data replication for distributed applications with examples of configurations supported by Replication Server. We then discuss the business and technical problems that such configurations solve. Next, we describe in detail how Replication Server works with emphasis on the benefits of the Sybase architecture. The Appendix to this paper describes in detail what needs to happen at each site when a replication system is put together.

Sybase Replication Server: A Practical Architecture for Distributing and Sharing Corporate Information

Sybase Replication Server, a key component of Sybase's Adaptive Server Enterprise product family was designed specifically to meet the needs of enterprise wide, client/server computing. Replication Server forms the foundation for data distribution in enterprise-wide client/server applications worldwide. This paper describes the unique advantages of using Replication Server to distribute and share corporate information.

Introduction

Replicating business data across an organization involves much more than simply copying a piece of information from one site to another. A replication system must address all of the following business needs:

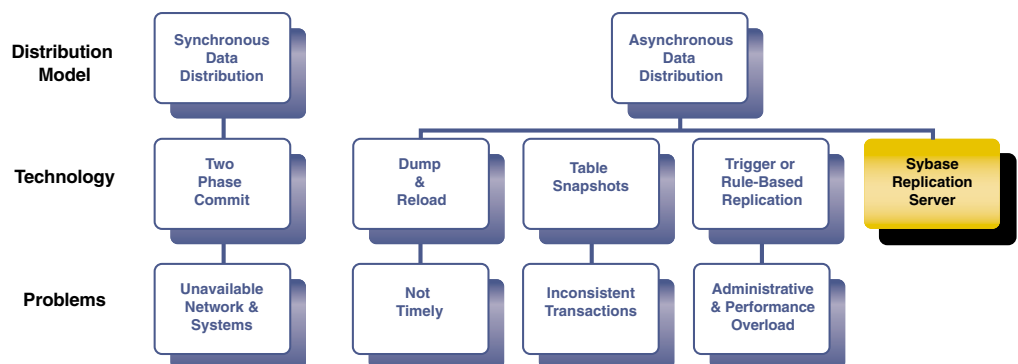
1. High Availability of Data: The replication system should be reliable and should not expose business operations to computer system failures.
2. Consistent Information Delivery: The distribution system must protect the integrity of the data.
3. High Performance: The replication system must not burden the source of the data, must use networks efficiently, and must allow each site to optimize access methods to local data.
4. Easy Centralized Administration: Administrators must be able to easily manage all the distributed components of the replication system from a single desktop.
5. Heterogeneous Data Source Access: The replication system should be able to move data across different vendors' data sources.
6. Local Autonomy: Each site which receives replicate data should be free to decide which set of data it wishes to receive, how it will view the data, how it will access the data and how it will modify the data.

As we will show in this paper, the practical architecture of Sybase Replication Server is ideal for distributing and sharing corporate information. The paper begins with an introduction to the concept of data replication for distributed applications, with examples of configurations supported by Replication Server. We then discuss the business and technical problems that such configurations solve. Next, we describe in detail how Replication Server works, with emphasis on the benefits of the Sybase architecture. The Appendix to this paper describes in detail what needs to happen at each site when a replication system is put together.

What is Data Replication?

Only a few years ago, corporate data resided in a central location. Remote departments accessed the information they needed by establishing direct connections to the central sites, or by requesting printed reports from central MIS. The connections however were expensive, unreliable, and limited in number, while the reports were inflexible and not timely.

Open systems brought inexpensive and powerful computing resources to all corners of an enterprise. The ability to share corporate information effectively using these new resources became an important competitive advantage for organizations. The question enterprises face today is not "Why distribute and share corporate data?" but rather, "How can one distribute information effectively?". As we shall show in this paper, replication is quickly becoming the architecture of choice for a majority of distributed corporate applications.



[Figure 1.] Data Distribution Alternatives

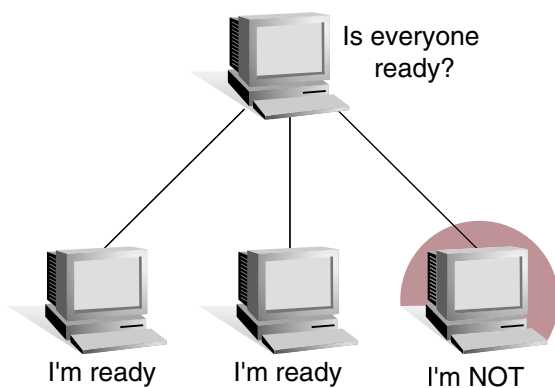
Early Data Replication: Dump and Reload

The concept of replicating data across an organization is not new. Organizations have been distributing copies of their data by "dumping" (downloading, perhaps to tape) and then "reloading" the data on a different site for years. It is not uncommon to see corporate data distributed across divisions by shipping a data tape from one site to another (a communication method commonly referred to as 'sneakerware'). Business units then make decisions based on data that may be days or weeks old.

As a result of "dump and reload" distribution, data from one location can be made available to other locations. The data at each site however, is not timely. Furthermore, the process is often manual and not automated.

Synchronous Replication: Two-Phase Commit

The introduction of two-phase commit technology in the late 1980s, which Sybase pioneered, provided a more timely way of distributing and sharing corporate data. Two-phase commit allows the synchronization of distributed data. A transaction will only be accepted if all interconnected distributed sites agree to it. An elaborate 'handshake' mechanism across the network allows distributed sites to coordinate their acceptance of each transaction.



[Figure 2.] In a two-phase commit setup, a transaction will not be accepted unless all interconnected sites agree to it.

While two-phase commit may be appropriate for situations where a corporation absolutely needs to synchronize distributed data, it comes at a price. Since all distributed sites need to synchronously approve a transaction before it is accepted, the corporate information system is exposed to individual component outages. If any one site is unavailable, the transaction will have to wait. Operations are therefore exposed to individual component outages. Furthermore, the elaborate handshaking mechanism, with messages going back and forth between sites as they coordinate the acceptance of the data, puts a significant burden on corporate networks.

Since network connections and individual components in a distributed system do fail, many organizations began to look for more practical and inexpensive ways of sharing corporate data with less exposure to their operations. It became clear that timely distribution of recent copies of data across an organization would address their problem. With a copy of the data available on site, individual locations no longer need to worry about the availability of networks or remote sites. They can continue their operations with the local copy of the data. The trade off, of course, is that the distributed sites need to be able to work with data that is not synchronized or real-time.

Non-Transaction-Based Replication: Snapshots

Early efforts to address this demand for distributed copies of data used table snapshots. Table snapshot implementations allow the asynchronous distribution of changes to individual tables, subsets of tables, or even collections of tables according to a pre-determined schedule.

While more advanced than "dump and reload" procedures, the use of table snapshots has several significant drawbacks that limit the applicability of the technology. Snapshots do not maintain the integrity of transactions and only provide 'read-only' copies that cannot be updated by the business units that access them.

Snapshots copy individual data tables or data items without maintaining the atomicity of a transaction. With transactions broken, the integrity of the distributed data is threatened. For example, if your bank used snapshots to distribute information on your bank accounts, it would copy your savings account information separately from your checking account information. If you transferred money from your savings to your checking account, the change to your savings account would not be copied at the same time as the change to your checking account. While gamblers may be attracted to such an uncertain banking arrangement, it is clearly unacceptable to any enterprise that values the integrity and consistency of its data.

Moreover, the use of table snapshots only provides a one-way path for data. While data can be copied to multiple locations, the copies are read-only and no changes can be made to the distributed data.

Non-Transaction-Based Replication: Database Triggers

In addition to snapshots, triggers are another asynchronous mechanism provided by some database vendors for the purpose of one-way data replication. In early trigger-based replication systems, customers were expected to assemble their own replication applications using these triggers.

You can think of a trigger as an alarm in the database that is associated with a specific piece of data. When a change is made to the tagged data item, that change 'triggers' an alarm inside the source database. The alarm in turn activates the replication-specific code inside the source database that begins the replication process.

Database triggers were originally introduced by database vendors to protect the referential integrity of corporate data and to centrally enforce business rules. The idea was that the database itself should include a mechanism to detect inappropriate data entries. Before the introduction of triggers, such filtering had to be built into all the applications that accessed the database. For database vendors who supported triggers, trigger-based replication seemed to offer a straightforward and simple extension for their products.

While offering customers more flexibility than snapshots, early trigger-based replication systems did not overcome the fundamental flaw of snapshot technology; the lack of protection of the transactional integrity of customer data. Trigger limitations include:

- Triggers simply transfer individual data items when they are modified, they do not keep track of transactions.
- Triggers allow one-way replication only. Data on replicate sites is read-only and cannot be modified.
- The execution of triggers within a database imposes a performance overhead to that database.
- Triggers require careful management by database administrators. Someone needs to keep track of all the 'alarms' going off when data is modified.
- The activation of triggers in a database cannot be easily 'rolled back' or undone.

To summarize, in early trigger-based replication systems, it was left entirely to the customer to build applications that kept track of and protected the integrity of transactions.

Transaction-Based Replication with Triggers or Rules

As we mentioned above, the extension of database trigger technology (or rules, which are a form of triggers) for the purpose of data replication offered database vendors a simple way to extend their lists of supported features.

Quickly however, vendors offering trigger or rule-based replication realized that they needed to address the fact that simple triggers or rules do not protect the transactional integrity of replicated data and do not allow changes to replicate data. Their customers needed flexible replication systems that protected the integrity of transactions.

For vendors who had started down the path of trigger or rule-based replication, the solution to this issue seemed obvious: Instead of letting the customers build their own replication systems using triggers or rules as their assembly tools, why not use the triggers in-house to build replication products? Vendors could then isolate customers from the underlying triggers or rules.

With the introduction of processes that grouped data changes into transactions after they were triggered inside the source database, trigger or rule-based replication systems solved the problem of losing the transactional integrity of the data. The filtering process, though, inevitably imposed a performance overhead.

While putting triggers and rules 'under wraps' may have solved some problems such as transactional integrity, it did not address several others, such as the performance burden that triggers impose to the source of the data and the administrative burden to system managers. As we mentioned above, these problems include:

- The execution of triggers within a database imposes a performance overhead to that database.
- Triggers require careful management by database administrators. Someone needs to keep track of all the 'alarms' going off when data is modified.
- The activation of triggers in a database cannot be easily 'rolled back' or undone.

Triggers impose a performance and administrative overhead to the source of the data. Such overhead is necessary when triggers are used to protect the integrity of data or to enforce business rules. However, the overhead is unnecessary and, in fact, inappropriate for the implementation of a replication system.

The trigger-based replication system is thus closely tied to the operations of the source database. Code needs to be executed within that source database in order to begin the replication process. Consequently, triggers and rules impose an unnecessary performance burden on the source of the data.

Sybase recognized early on the limitations of trigger-based replication and chose to implement a flexible replication architecture that does not burden database administrators and system performance with triggers.

State-of-the-Art Replication: Sybase Replication Server

After seeing some of the early replication alternatives listed above, organizations quickly discovered that a reliable replication system must do much more than simply copy a piece of data. The system must also:

- Maintain the integrity of the data at the transaction level
- Deliver data quickly and efficiently across the network
- Allow distributed sites to modify data
- Be easy to monitor and manage (the most important, perhaps)
- Transfer data in any direction across heterogeneous data sources

The introduction of Replication Server in the fall of 1993, brought the first and to this date the most sophisticated such product to the market. Working closely with its customers, Sybase created a new, practical, and reliable alternative for the distribution of corporate information.

Replication Server introduced several important new elements to the concept of data replication. The asynchronous operations of the product can be divided into six components:

1. Transactions are detected and captured by the replication system in one or more source databases where changes are being made with minimal performance overhead. No triggers or rules are used for this process.
2. The information is then distributed across local or wide area networks in a manner that allows administrators to make the best use of their network resources.
3. Transactions are delivered to their ultimate destinations without limiting the flexibility of autonomous users at these destinations to customize or modify the replicated data.
4. A powerful graphical systems management tool allows administrators to monitor and manage every component, or collections of components, of the replication system from a single central site.
5. If a network or system component fails, data in the process of being delivered is stored temporarily in disk queues; when the component that failed returns to operation, replication restarts and the queued data resume their delivery path.
6. Using Sybase products, customers can build applications that replicate data changes in any direction across different hardware platforms and heterogeneous data sources.

Where Does One Use Replication Server?

Before we cover in more detail how the Sybase Replication Server works, we continue with some examples of how corporations can use Replication Server in their distributed enterprise environments and for high availability. The list is not exhaustive, but illustrates some of the benefits that a flexible replication architecture can offer in a distributed environment.

As a key component of a cost-effective distributed system, Sybase Replication Server addresses the following critical issues:

- **High Availability/Disaster Recovery.** Replication Server delivers high availability by propagating data to multiple locations. Systems remain online despite hardware, software and network failures. Replication Server maintains a near real-time "warm standby" database to which applications can switch with virtually no downtime if the primary site fails. Warm Standby systems can be configured over Wide Area Network and guarantee the integrity of the database.

Businesses today are faced with the critical need to ensure the availability and continuous operation of their business systems in spite of potential failures ranging from disk crashes and CPU failures to catastrophic losses of their computing facilities or communications networks, and planned downtime for maintenance. Many companies also have geographically dispersed data centers, that need information available across those data centers, particularly when a data center has an outage, planned or unplanned. While off-site tape dumps have traditionally satisfied the requirements for disaster recovery for batch systems, they are typically inadequate for protecting the information in On-line Transaction Processing (OLTP) systems and e-business. Asynchronous replication facilities can provide continuous duplication of critical OLTP and e-business application information to off-site backup facilities without the high latency inherent in tape backup strategies. Once established, such an environment can be automated to ensure that information is replicated in a timely manner and the switch to backup systems is accomplished with minimal business interruption. Replication Server can replicate DDL and schema changes, and can be configured to replicate the entire database, thus creating a complete mirror of the database.

High Availability should address availability of data in three areas:

1. Manage Planned Downtime (e.g., routine maintenance downtime, software upgrades, etc.).
2. Protect during Unplanned Downtime (machine/network outages).
3. Provide Disaster Recovery. Any system providing HA services should provide continuous availability of data in all three scenarios.

Replication Server offers data availability in all three scenarios by providing a standby system.

- **Distributed Systems.** Replication Server allows an organization to locate data where it is needed, making distributed business units much less susceptible to central computer or network downtime while reducing overall communications costs. It allows bi-directional data sharing with a safe approach for replicating remote updates. And it allows a corporate overview of distributed operations that is very close to real time, even when distributed business units run on a variety of hardware and DBMS platforms. Replication Server helps companies find the balance between centralization and decentralization. It allows data to become a corporate asset that is stored centrally or distributed. Data replication is a tool that helps companies put necessary data in the hands of local decision-makers and also maintain firm central control over the data.
- **Live Decision Support.** Replication Server can replicate an OLTP database, this allows users with query intensive requirements to utilize data that is within seconds of real time, without affecting OLTP performance.

Depending on the design of the distributed system that uses Replication Server, we can distinguish three types of sites in a distributed system:

1. **Primary (or sites with primary data only):** Locations from which data is replicated to other systems. Primary data can be modified on these sites.
2. **Secondary (or sites with replicate data only):** Locations that receive data from one or more primary sites. The replicated data is read only and cannot be modified.
3. **Peer-to-Peer (or hybrid sites with primary and replicate data):** Locations from which data is replicated to other systems, and which receive data from one or more other primary sites. Peer-to-peer replication environments may involve applications where update conflicts are avoided or applications where update conflicts are resolved.

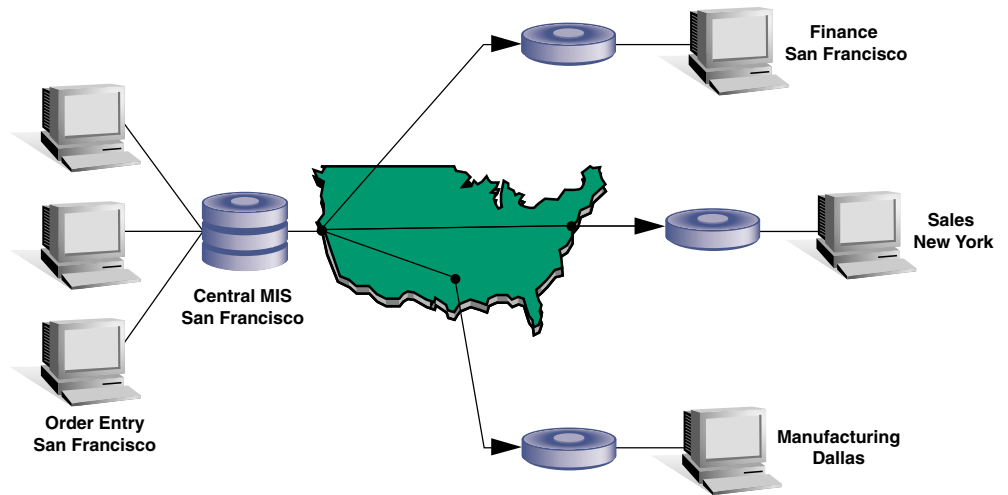
As we shall illustrate, Sybase Replication Server may be used for applications that include any combination of these types of sites. Information may be replicated to and from Sybase or non-Sybase data sources.

Example 1: One Primary - Multiple Secondary (Decision Support Replicates)

In its simplest use, Replication Server provides an inexpensive way of distributing information from a central site to many replicate sites where the information will be read only and will not be modified. Unlike snapshots, the distribution mechanism maintains the transactional integrity of the data.

Figure 3 illustrates the scenario: A company uses a central OLTP database (which may in fact reside on a legacy data source) in San Francisco to process incoming orders. Order entry applications are connected directly (as terminals or clients) to the central OLTP system that is controlled by corporate MIS staff. A large number of decision-makers in other parts of the organization (Finance which is also located in San Francisco, Manufacturing in Dallas, Sales in New York) would like to view order information to make timely decisions in their respective organizations. However, due to the taxing nature of their ad hoc queries or their customized batch reports, or simply due to their large number, they are currently not allowed to access the OLTP system. These decision makers thus have to look for the information they need in standardized reports generated by the MIS staff.

If Replication Server is used, individual divisions can 'subscribe' to a subset of the central site's data and view that information locally. With a local copy of data needed for decision support, Finance, Sales, and Manufacturing can carry out their operations without having to login to the central MIS site in San Francisco. Their decision support work is consequently isolated from network or remote system outages. Furthermore, local users are likely to receive the results of their queries faster, since they are accessing local (replicated) rather than remote data.



[Figure 3.] *Decision Support Replicates*

Thus, replication can provide a simple way to increase the scalability of a system by providing data access to more corporate decision-makers. As we shall discuss later in this paper, Replication Server is architected so that the replication process does not burden the source database. As long as the subscriptions on the replicate sites are small enough to allow the replication system to keep up with the activity in the OLTP database, Replication Server provides an excellent way to increase the scope of an existing OLTP system. Given the flexibility offered by Replication Server, the primary data may reside in any Sybase or non-Sybase data source, including legacy data sources. More information on the heterogeneous data source support capabilities of Replication Server is included later in this paper (see 'How Does Replication Server Work' section).

A variation on this example is to use a warm standby database as a decision support database. The following example provides information on warm standby environments.

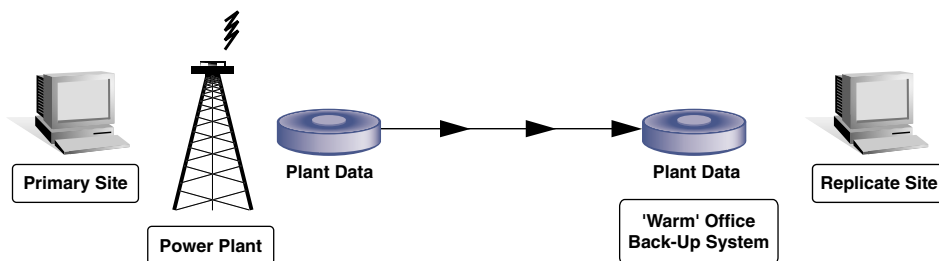
Example 2: Warm Standby Database

Replication Server provides a cost-effective way of setting up near real-time standby copies of a database. This warm standby copy can then be used in the event of failure at the primary site, with user applications all 'switching' over to the standby site. The Replication Server provides a warm standby capability that replicates an entire database to a remote ASE server. Not only is data replicated, but data definition language (DDL) syntax such as create/drop tables is also replicated in a warm standby environment. This provides an exact copy of the primary database at the replicate site.

Figure 4 provides an example: A large electric power company has a database system on site at the central power plant that controls the production and transmission of power from the power plant to individual customer homes. The application that controls power transmission is clearly a mission-critical one where a redundant architecture is required. Data is mirrored to multiple disks on site; network connections are redundant, etc. There is one problem, however, in the event of a major catastrophe, such as an earthquake, a flood, or a fire at the power plant, the system which controls power switches and transmission lines would be unavailable. The company's redundant architecture requires that a standby system be maintained off site. In case of a catastrophic event at the primary (power plant) site, system administrators could switch applications to the off-site, redundant system. After switching, any changes made to the warm standby server can be replicated back to the primary server. The Replication Server will store the transactions and apply them to the primary server, once the server recovers. This provides a "fail-back" capability where the primary server will be re-synchronized with transactions that were applied to the warm standby server.

This same mechanism can also be used for planned downtime such as system maintenance, where information can be replicated to a local or remote warm standby site. This standby database can be used, updated, etc. while maintenance is being performed on the primary database, and then the data (with all the changes that have taken place on the warm standby database) can be 'switched' back when the primary database is available.

Finally, the warm standby server can be used for decision support since it is an exact duplicate of the primary server. The standby database provides management with power usage statistics and customer power consumption in near real-time without impacting the primary database.



[Figure 4.] 'Warm Stand-by' Database

There is an important trade-off that one must consider when setting up a redundant primary. Since a replicate copy will lag the primary copy under normal conditions by a few seconds, this lag or 'latency' must be allowed for in the design of the fail-over application. Transactions that have not been sent to the Replication Server will not appear in the warm standby database. However, any transactions that have been sent to the Replication Server will be forwarded to the warm standby database. In the example provided above, a few seconds worth of changes made to the primary site may not be available at the replicate site after the fail-over. Clearly, however, in this case the electric utility benefits by having the option to control its power transmission lines from a location that is remote to the power plant.

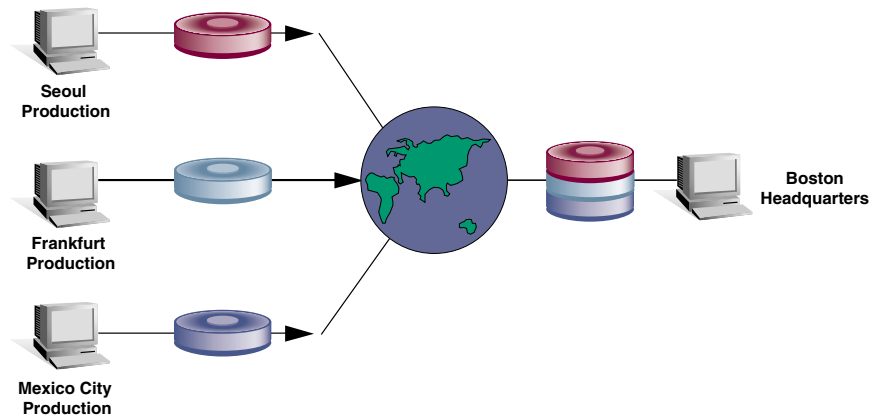
It should also be pointed out that the standby copy would inevitably fall behind if an entire database with high transaction rates were to be replicated. System administrators need to take into account the throughput of the replication system when setting up a warm standby system.

Example 3: Multiple Primaries - One Secondary (Corporate Data Consolidation)

In addition to distributing information from a central location, Replication Server provides a simple mechanism to bring together corporate information from several locations and to consolidate it in one site.

One scenario for such a setup is a corporation that gathers information from remote production sites for central analysis at corporate headquarters. Figure 5 illustrates a possible example: A multi-national company has several facilities distributed worldwide, with production sites in Seoul, Frankfurt, and Mexico City. Corporate headquarters are in Boston. Boston headquarters needs to have an up-to-date, read-only view of production status worldwide.

Consequently, information from the three production sites is consolidated in Boston. While the data in Boston may be a few seconds or minutes old, it is near real-time and current enough for the corporate applications in Boston. Replication Server provides a cost effective way for the corporate consolidation of distributed data. Note that, as we shall discuss later in this paper, the database schemas in Seoul, Frankfurt, Mexico City, and Boston may be different.



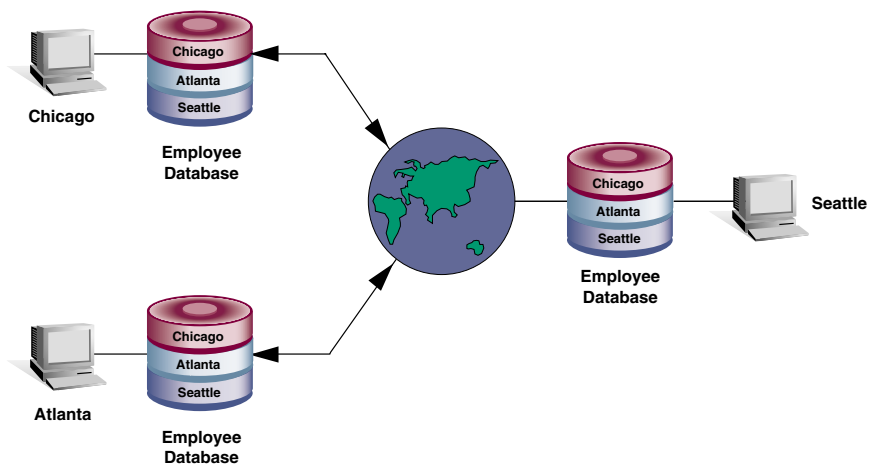
[Figure 5.] Corporate Data Consolidation

It is worth comparing this configuration to 'dump and reload' applications that currently exist in many business structures. Rather than receive from each site data tapes or reports that are days or weeks old, Boston receives production data that is only a few seconds old.

**Example 4: Peer-to-Peer Configuration With No Update Conflicts
(Corporate Data Sharing)**

The two previous examples described cases where replication occurred in a single direction, in the first case from one site to many, and in the second example from many sites to one. Replication however does not necessarily need to involve data transmission in a single direction. Corporate Data Sharing provides one illustration of this. Under this configuration scenario, shown in Figure 6, several distributed systems are setup so that they include primary as well as replicated data.

Here's an example: A company with several sites across the US (Atlanta, Chicago, Seattle) wants to consolidate its employee data. Each employee has one (and only one) home office. Each regional office manages local employee information. Replication Server provides a mechanism for the corporation to share this distributed set of employee data.

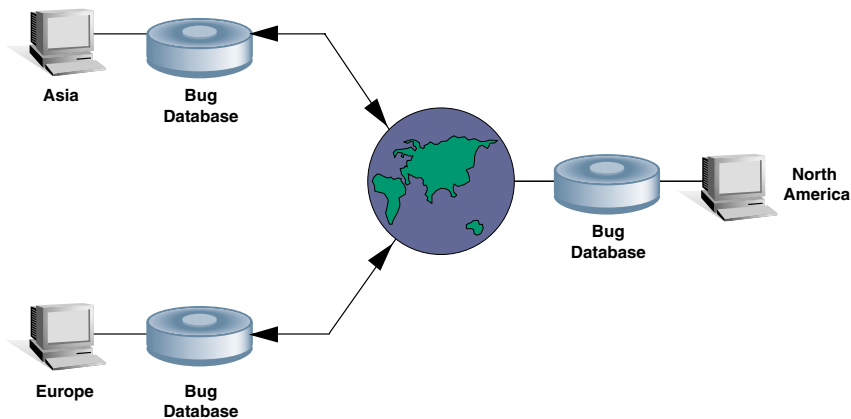


[Figure 6.] Corporate Data Sharing

Since each employee has only one home office, the example illustrates the case where distinct data items are combined from each site and where the distributed application does not involve update conflicts. While each site includes primary as well as replicate data, no two sites are allowed to modify information on the same employee. Thus, the Atlanta office will only modify information on Atlanta employees; the Chicago office will edit information on Chicago employees and so on. The result is that all sites can view collectively near real-time employee information for the entire corporation.

Example 5: Peer-to-Peer Configuration With Update Conflicts (Corporate Data Integration)

The previous example focused on the combination of distinct data items from several sites where no two sites can modify the same piece of data. While it is always simpler to design and manage an application that avoids update conflicts altogether, conflict avoidance will not always be possible. In many situations business operations will require that more than one site modify a common data item. Figure 7 illustrates such a distributed update example.



[Figure 7.] Corporate Data Integration

A multi-national software company has support analysts in three locations, one in Asia, one in Europe and one in North America. Support analysts in each location respond to calls and assist regional customers. All analysts share a common bug database, where information on customer problems is logged. The bug database needs to be accessible by all three sites. Furthermore, analysts in all three sites need to be able to enter and modify information in the bug database. This is a case where data items (i.e. bug information) may be updated by analysts from any support site. Since more than one site can update the same data item at the same time, the support organization's bug tracking application will need to include an update conflict resolution mechanism.

A more detailed discussion on Sybase recommendations for update conflict resolution in environments similar to the one discussed here can be found in the 'How Does Replication Server Work' section of this paper. Sybase recommends an approach that makes such a sophisticated distributed environment manageable.

Example 6: Upgrade an ASE Application

Upgrading a production system to a new version of a data server is a difficult and potentially dangerous task. New features, or changes in functionality can cause unexpected down-time. Applications must be fully tested with the new version before it is rolled out into the production environment. Additionally, existing applications will not take advantage of new features in the ASE server. Warm standby replication is a fast and effective way to test a new version of an ASE server.

In this example, an MIS department is responsible for upgrading the ASE server that houses the database for a business critical inventory system. The database administrator (DBA) wants to upgrade the ASE server to take advantage of new performance features. The DBA installs the new version of ASE on a test computer system and creates a warm standby environment using Replication Server. All transactions including all DDL are replicated to the new ASE server without impacting the production application. The new version of the server is literally tested by the production application but in the test environment. Any issues with the new version will be identified and fixed before it is rolled into production. Once the new server is stable, the DBA can test the new performance features. When the DBA is confident that the new version of the server is ready for the production system he can upgrade the production ASE server. However, because the Replication Server keeps the production and test databases synchronized the DBA can simply switch the test and production computers and automatically upgrade the ASE server.

Example 7: Non-Sybase Data Replication

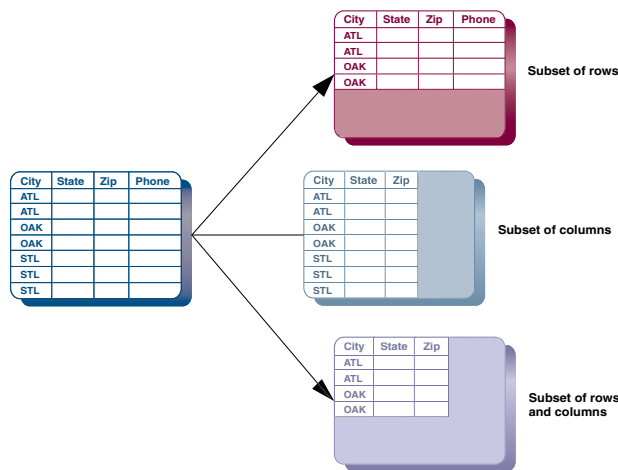
Sybase Replication Agents and DirectConnect™ gateways provide replication to and from non-Sybase databases such as DB2, Oracle, and Microsoft SQL Server. Data in legacy systems can be replicated to smaller, departmental size systems without interrupting or re-coding legacy application. Figure 18 in the Appendix illustrates the configuration for replication from a non-Sybase database.

For example, a large corporation has an OLTP system running in DB2 on an IBM mainframe computer. The system has been in production for years and is critical to the business. Upper management needs information from the OLTP system for long-range planning and statistical analysis. The DBA installs the Sybase Replication Agent for DB2 on the IBM mainframe, and sets up a Replication Server and ASE server on a small departmental computer. The required data is replicated from DB2 to the ASE server where it is consolidated and merged with data from other parts of the corporation. The OLTP system is not impacted and does not have to be modified. The mainframe department does not have to download data to tape, or produce reports for management. The data is sent to the replicate database in almost real-time providing up-to-date information for better decision-making.

Example 8: Replicate All or Part of the Table

As these cases illustrate, the use of replication involves a trade-off between data availability versus data synchronization. Replication provides a cost-effective way of reducing operational exposure to component failures at the cost of working with near real-time rather than synchronous information. As we shall demonstrate in the next few sections, Sybase Replication Server provides the most efficient and safest architecture for implementing configurations such as the ones listed above.

In all of the examples above, data replicated may include an entire table, a subset of rows of that table, a subset of columns or both, as illustrated in Figure 8.



[Figure 8.] Replication Server allows the replication of selective subsets of tables.

Each site has the freedom to decide which data to replicate and how it will view the replicated data.

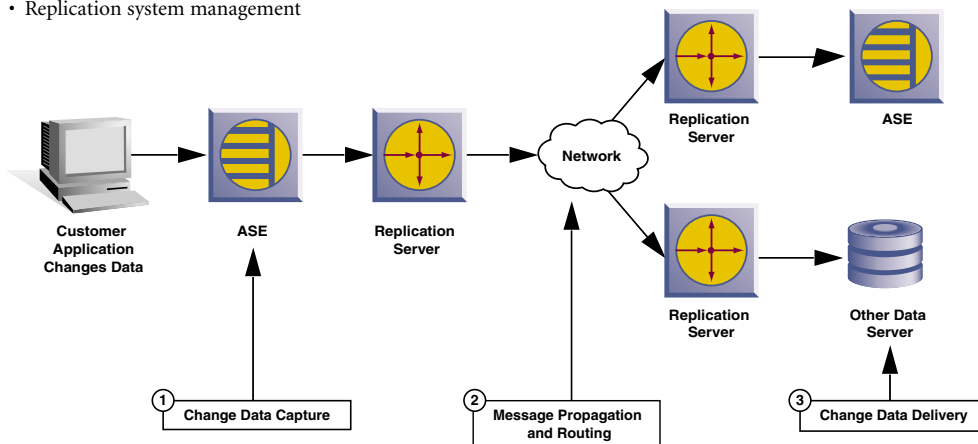
Having outlined what Replication Server can do, let's now discuss briefly how it works.

How Does Replication Server Work?

We now introduce the components of the Sybase Replication Server product and outline how they function.

We can divide the operations of a replication system into four areas (see Figure 9):

- Transaction (data change) detection from one or more source databases
- Transaction propagation and routing across the network
- Transaction delivery to the target databases
- Replication system management



[Figure 9.] Sybase replication system components (in one-way replication setup)

In a Sybase Replication Server environment, a process running inside the ASE called the Replication Agent™ thread detects modifications to data in source databases. The Replication Agent thread then passes the changes onto a Replication Server process, which may be on a separate system. A collection of Replication Servers, which are typically on separate LANs, then cooperate to direct data changes to their destinations using pre-configured intelligent routes. In the final step of the replication process, a Replication Server delivers the data changes to the target databases. The entire operation is managed and monitored from a single desktop using the GUI-based environment of Replication Server Manager™. Data on its way from a source database to its destination is protected from brief component failures with the use of stable queues. Stable queues are a safety mechanism that provides the replication system with a configurable level of tolerance to individual component failures.

Transaction (Data Change) Detection – Replication Agents

The Sybase Replication Server product includes a component called the Replication Agent. The Replication Agent process executes as a thread inside the ASE. The Replication Agent detects transactions that contain data modifications and forwards them to the Replication Server. The Replication Agent is responsible for a specific database. If multiple databases are the source of replicate data, then multiple Replication Agent threads will execute inside the ASE.

The Replication Agent thread reads the primary ASE database transaction log to detect changes to the data. The transaction log is the best source of information about primary data modifications, because it contains records of committed, recoverable transactions. In Figure 10, when an application makes a change to data at the primary database, that transaction is recorded in the transaction log and, to protect the integrity of the data, written to disk at commit time. The replication process in no way interferes with this critical function of the database system. The Replication Agent monitors the activity of the ASE database transaction log and when it detects a transaction for a table or stored procedure that has been marked for replication, it passes it on to the Replication Server.

The Replication Agent can connect to a Replication Server that resides on a separate system. Administrators can minimize the overhead that the replication activity imposes on the system where the primary database resides by having the Replication Server process on a separate system.

The Replication Agent process translates ASE transactions to data source independent commands understood by the Replication Servers. For example, if an update is taking place in a transaction, the Replication Agent will translate the update command to an rs_update command and pass it on to a Replication Server process. This translation at the source database to commands understood by the Replication Servers is an important strength of the Sybase replication architecture. The interface between the Replication Agent and the Replication Server called the Log Transfer Language (LTL) is visible to customers. This data source independent translation allows the customer to manipulate the data in a transaction.

Sybase also provides Replication Agents for non-Sybase data servers such as Oracle, DB2, and Microsoft SQL Server. These Replication Agents capture changes in the Non-Sybase servers by either monitoring the transaction logs or by using database triggers. The Replication Agents translate the commands to LTL and sends them to the Replication Server. This allows companies with legacy systems to replicate data to open systems without modifying or impacting existing data sources or applications. Finally, because the LTL is visible, customers can use Sybase Open Server™ technology to build custom replication agents for data sources not supported by Sybase.

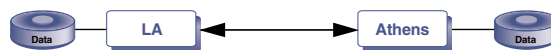
It is important to note here that the overhead and interaction of the Replication Agent (and consequently, the replication system) with the primary data source is minimal. As we shall show later, this attribute is an important distinction between the Sybase architecture and alternative replication models. Unlike trigger or rule-based replication architectures, the Sybase model with Replication Agents in no way interferes with the normal operations of the primary databases.

Data Propagation and Routing Across LANs and WANs

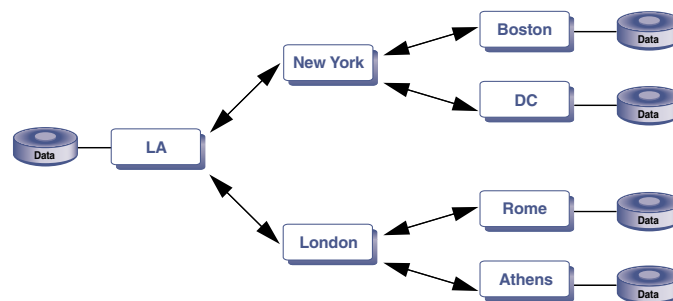
Let's now look at what happens after information is transferred from the Replication Agent to its associated Replication Server. If the source (primary) data items and the target (replicate) data items are on separate LANs, the next step in the replication process is the propagation of data from the Replication Server(s) associated with the primary database(s) to the Replication Server(s) connected with the target database(s).

Note that if the source and target databases are both on the same LAN, there may be no need for a second Replication Server. In a single-LAN replication environment, one Replication Server may be sufficient to coordinate the delivery of messages to and from data sources throughout the LAN (Figure 15 in the Appendix illustrates this point).

In the multi-LAN scenario, routes between different Replication Servers may be direct (as illustrated in Figure 10a), with no intermediate sites between the primary and the replicate, or indirect (Figure 10b), with one or more intermediate sites between primary and replicate.



[Figure 10a.] An example of a direct route between Replication Servers in LA and Athens



[Figure 10b.] An example of indirect routes between Replication Servers

The direct or indirect routes that replicated data take are pre-configured by systems administrators. Routes allow administrators to make the most efficient use of corporate networks in accordance with the constraints of their networks and the data delivery requirements of their applications. Configuration and status information on direct and indirect routes can be monitored from a central location with the Replication Server Manager.

Data Delivery to Sites with Replicate Data

In the previous two sections we examined how the Replication Agent and the Replication Server processes work together to detect and extract data changes from primary databases and to deliver them across the network toward their destination. We now examine the final component of the replication process, the delivery of replicated data to one or more replicate sites.

As we mentioned earlier, the Replication Server process is itself a Sybase Open Server/Open Client™ application. In the final part of the propagation of data changes from a primary database to a replicate site, a Replication Server establishes a connection to the database where data is to be delivered. The database may be a Sybase or a non-Sybase system.

The Replication Server connects to the target database as a privileged client user in a standard client/server connection. It then delivers the replicated data to its destination. Later in this paper, we will describe how Replication Server 'knows' which data to deliver and where. As we shall show, replicate sites 'subscribe' to the information that they need.

Since Replication Server simply delivers data to the target data sources like any other database client would, replicating data to non-Sybase data sources is straightforward. Customers can simply use Sybase DirectConnect to allow the Replication Server to connect to non-Sybase data sources. Conceptually, one can think of these gateways as a mechanism by which non-Sybase data sources appear (to application clients) as ASE servers. Consequently, no changes need to be made to the Replication Servers in order to replicate to non-Sybase data sources, accessible via such gateways. However, Replication Server does provide data translation vehicles called function strings and user-defined datatypes that the customer can use to change the data within a command or even change the actual command sent to the replicate data server.

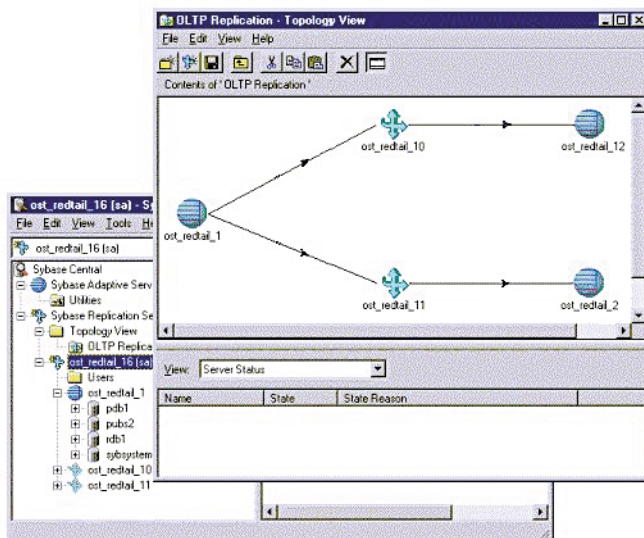
Since the ultimate target of the replicate data is a regular ASE or non-Sybase database, local users are free to customize their database access methods for maximum performance just as they would for any local non-replicate data. The replication system does not restrict local sites from customizing indexes, data distribution on disks, etc. for optimum data access performance.

Replication System Configuration and Management

Clearly, a powerful systems management tool is essential for the successful implementation and administration of a large distributed replication environment. Sybase realized this fact early on in the design of its Replication Server product and built the Replication Server Manager (RSM). RSM is a powerful three-tier management tool with a graphical user interface where system administrators can create replication environments, monitor the status of the individual servers and manage the flow of replicate data, all from a single desktop.

Servers in the replication environment such as the ASE servers and Replication Servers, appear as icons on the RSM screen. The RSM desktop, illustrated in Figure 11 allows administrators to manage and monitor not only individual components but, most importantly in large distributed environments, collections of components.

The example displays the RSM desktop and topology windows. RSM is a plug-in on the Sybase Central™ desktop. The topology window displays the servers and connections in the replication environment. The user can customize the topology window to display any set of servers. The status of the servers and connections are displayed on both the RSM desktop and topology window.



[Figure 11.] Replication Server Manager™ screen shot

Features provided by the RSM include:

- Creating the connections, routes, replication definitions, publications, and subscriptions needed to define the replication environment.
- Creating and monitoring the Replication Server stable queues.
- Monitoring the status of the servers in the replication environment and visually displaying their state on the RSM desktop.
- Viewing the server's log file.
- Controlling the flow of data through the replication environment by suspending and resuming connections and routes.
- Graphically displaying the latency time, or data delivery time between the primary and replicate databases.
- Assisting the administrator when replication errors occur by displaying data exceptions and transactions within the stable queues.
- Automatically notifying the administrator when a replication event occurs such as when a latency threshold is reached. These events and the notification procedures are configurable by the administrator.

Replication Server Manager brings the sophistication of object-oriented distributed systems management technology to the replication environment. A replication environment would be difficult to manage without a tool as powerful as the RSM.

What if a System or Connection is Unavailable: Stable Queues

Sybase's modular replication design allows the replication system to resume operations quickly and easily after brief network or system component failure. This configurable protection of the replication system from component failures is achieved through the use of disk queues.

Each step in the replication process stores the data on disk so that no data is lost if the next component is unavailable. The Replication Agent keeps track of the last transaction successfully sent to the Replication Server. If the Replication Server is unavailable, data stays in the database's transaction log. The Replication Agent begins sending the data when the Replication Server becomes available. Similarly, Replication Server stores transactions in stable queues so that no data is lost if a replicate data server or another Replication Server is unavailable. Transactions are only removed from the queues when they are successfully sent to the next component. Therefore, if a data server or a Replication Server goes down, no data is lost. When the server becomes available again, the Replication Server will start sending the data stored in the queue to the replicate sites.

The stable queues also allow the Replication Server to handle high volumes of data. Loading and unloading the stable queues happens independently so during peak times, more data might be flowing into the queues than is being extracted from the queues and sent to the replicate data servers. However, once the data load is reduced, the Replication Server will catch up with the workload and finish replicating the data. Transaction order and integrity is maintained, and no data is lost.

The bottom-line is, system administrators can configure disk space where data in the process of being replicated can be queued temporarily while a system component or network connection is unavailable. Replication resumes automatically and the queues are emptied when the component returns to operation.

Modifying Replicate Data: Distributed Updates

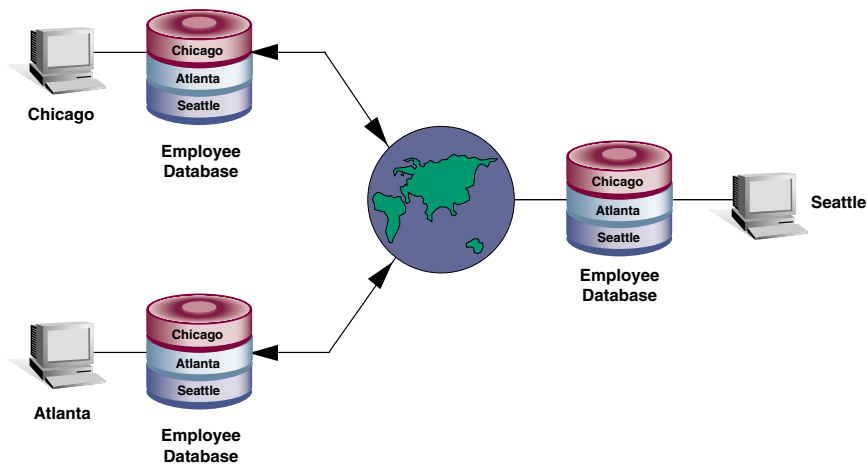
What if users at replicate sites want to make a change to the data? As we discussed in the Corporate Data Integration example above, there will often be situations where the same data item needs to be modified by users on more than one replicate site. Let's examine how users can accomplish this business need with Replication Server.

In a Sybase Replication Server environment, updates to replicate data can occur in two different ways:

- Asynchronously, by transferring procedure calls between sites.
- Synchronously, through direct connections to the system which owns each data item.

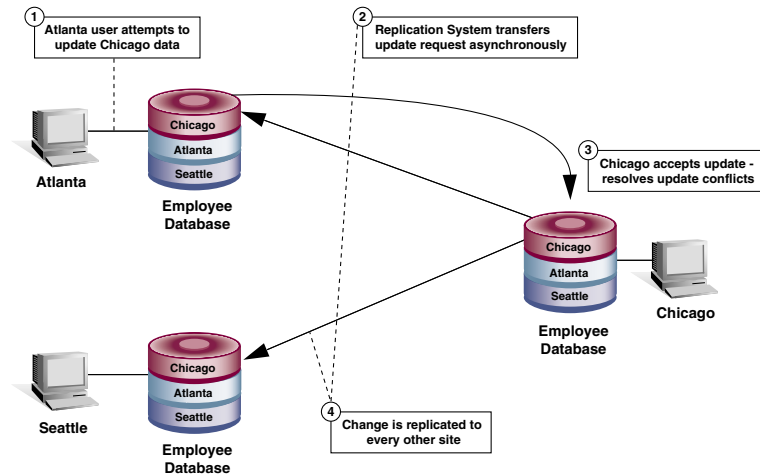
Sybase recommends that at any point in time each piece of data should have a clearly identified site that controls it. While updates can originate anywhere, each piece of data in the system has a site that 'owns' it at that time.

Let's look at an example: In Figure 6 we described a set-up where an employee database is shared across three locations, Chicago, Atlanta, and Seattle. It was assumed that Chicago employee information, although visible via replication to Atlanta and Seattle, would only be modified by users in Chicago. What if this assumption is not true. What if, for some reason, business operations require that someone in Atlanta needs to have permission to modify Chicago data. Can this be done?



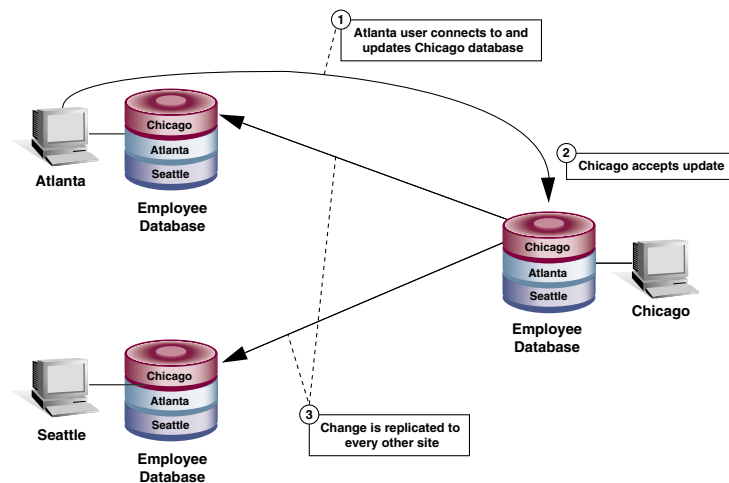
[Figure 12a.] Corporate Data Sharing

The answer is yes. As illustrated in Figure 12b, system administrators can set the replication system up so that an attempt in Atlanta to update Chicago employee information results in the asynchronous communication of an update request to Chicago. The replication system is used to transfer from Atlanta a request for an update to Chicago. Setting this up is as simple as defining a stored procedure for local updates. Once the change in employee data originating from Atlanta is accepted in Chicago, it is then replicated to every other site including Atlanta. Consequently, the Chicago site acts as a clearing house for all changes to Chicago employee data, even if the change was initiated in Atlanta or elsewhere. In a similar manner, the Seattle and Atlanta sites act as clearing houses for any changes to Seattle and Atlanta employee data respectively, regardless of the origin of the change. This allows the owner of the information to control access.



[Figure 12b.] An example of a distributed update using asynchronous procedure calls.

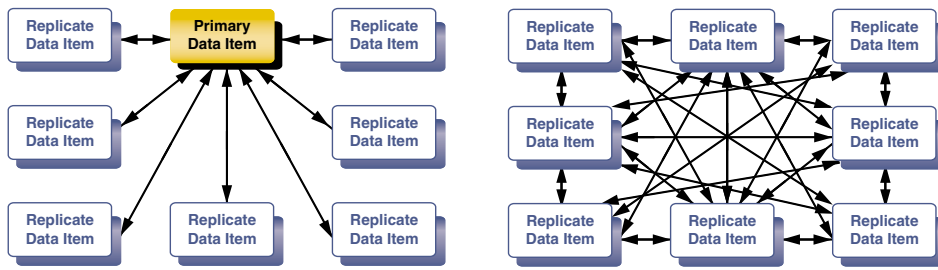
Another alternative for Atlanta users, illustrated in Figure 12c, is to actually log in to Chicago when they need to make an update to Chicago data. After connecting to Chicago and making a change there synchronously, the modification to the data is replicated to all sites.



[Figure 12c.] An example of a distributed update with a direct connection to the primary.

Why is ownership of a data item at any point in time important? Consider the alternative, illustrated in Figure 13. In an update anywhere scenario where data belongs to 'everyone', valuable system cycles everywhere will need to be spent in conflict resolution; furthermore, system back-up and recovery will also be far from easy. And last, but not least, it is an environment extremely difficult to design.

It has been suggested that users may be able to purchase and install a 'turnkey' conflict resolution mechanism that will coordinate distributed updates automatically. Clearly such claims with regards to 'generic' conflict resolution oversimplify the distributed application design complexities of a replication environment.



Sybase Remote Updates:

Async. Updates from Replicates
 Single 'Clean' Reference Point
 Low Overhead for Conflicts

'Turnkey' Conflict Resolution:

Different to Design
 Difficult to Recover after Failure
 'Overhead Everywhere'

[Figure 13.] A comparison of distributed update models for replication systems

The 'turnkey' conflict resolution model on the right hand side of Figure 13 is not manageable. At any given point in time data is 'floating' or being negotiated between sites making the distributed system extremely difficult to recover. It is very difficult at any point in time to identify what the 'real' value of a piece of data is.

Furthermore, conflict resolution is far from straightforward. If conflict resolution is done 'automatically' by the replication system according to certain rules such as update time (e.g. initial 'update' or latest 'update' has priority), or according to arrival time (e.g. last update to arrive has priority) simple questions one would ask are: Whose time? Each system typically has its own clock. Arrival time? Where, due to communication delays, data changes may arrive at each site in a different order. These simple questions reveal that the promotion of 'turnkey' conflict resolution oversimplifies the design challenges of building a distributed application.

To summarize, while there is nothing in the architecture of Replication Server that prevents customers from designing and implementing an environment where update ownership is shared between sites and where conflict resolution takes place everywhere, Sybase recommends that a reference point be identified for each piece of data, to make the replication environment manageable.

Why Sybase Replication Server?

In the previous sections we introduced the concept of data replication, explained how the Sybase Replication Server works, and provided several examples of replication environments. Having examined how this product works and where it can be used, we can now outline in retrospect the components of Replication Server that make the Sybase product superior to any other related offering in the market.

Replication Server Benefits

High Performance

High Performance in the context of data replication has three equally important parts:

1. **High Performance Transaction Detection:** The replication process does not burden operations at the source of the data to be replicated.
2. **High Performance Message Delivery:** The replication system uses the network efficiently for the distribution of replicate data.
3. **High Performance Data Access:** The replication system in no way limits the flexibility of local sites to optimize the access paths to the data they need.

High Performance Data Capture: No Triggers or Rules

The components that comprise the Sybase Replication Server are not features or functions of a database system, but rather a separate product designed specifically for replication. Replication Server components are clearly separated from the activities of a source database. The Sybase Replication Server architecture, which does not use triggers or rules, does not burden the source database. One of the reasons organizations choose to replicate data is to off-load the work that is performed on the primary server. Consequently a replication system that imposes a significant performance and administration burden on the source database would in part defeat the purpose for which it is introduced.

As we described in the introduction to this paper, in a trigger-based system a change made to a source database 'triggers' an alarm inside the database. The alarm, in turn, activates the replication-specific code inside the source database that begins the replication process. The trigger or rule-based replication system is thus closely tied to the operations of the source database. The code needs to be executed within that source database in order to begin the replication process. Consequently, triggers and rules impose an unnecessary performance burden on the source of the data. In addition to potential performance overhead, triggers and rules require careful monitoring. Database administrators need to keep track of all the 'alarms' going off when data is modified. The performance and administrative overhead is perfectly acceptable and in fact necessary when triggers or rules are used to protect the integrity of data or to enforce business rules. However, the overhead is unnecessary and, in fact, inappropriate for the implementation of a replication system.

Conversely, Sybase Replication Server provides a modular mechanism where the replication system is isolated from and does not burden the performance at the source of the data, or the operations of the database administrator responsible for the primary database.

High Performance Data Delivery

The Sybase Replication Server architecture allows administrators to use their networks efficiently by choosing the routes that replicated data will take across the network. Routes can be direct or indirect. While messages may reach their destination marginally faster via direct routes since they are shorter, indirect routes offer administrators great configuration flexibility.

An indirect route from San Francisco to Athens, Greece could send messages through London, with one telecommunications link between San Francisco and London and another between London and Athens.

Routes with intermediate sites have important advantages:

- Reduced WAN volume: Messages sent to an intermediate site can then be propagated to all downstream sites from that intermediate location. Consequently, indirect routes reduce the number of network connections and the number of messages sent across the network.
- Flexibility to distribute activity among Replication Servers: With the help of indirect routes, processing load related to replication can be shared among Replication Servers.
- The existence of indirect routes also allows administrators to switch message routes in case a particular WAN fails.

High Performance Data Access

The Sybase replication system delivers data to ASE and non-Sybase data sources. Each site is free to configure the local data source in accordance to local data access needs. The replication system in no way restricts the freedom to customize local schema design at each remote site. The replication system does not interfere with local data access method optimization choices, database index design, or data distribution on disks.

Consistent Information Delivery

Replication Server transfers transactions, not rows. The transactional integrity of replicated data is therefore protected across the replication system. The protection of transactional information distinguishes Replication Server from early replication technologies such as Table Snapshots. Furthermore, unlike trigger or rule-based replication systems, the Replication Server transfers the transactions themselves, not row changes grouped by transaction. While more recent versions of trigger-based replication products have succeeded in providing transaction-based replication, they do so by incurring significant performance overhead at the source of the data.

Since the Sybase Replication Server transfers transactions, it can also transfer stored procedures, thereby providing an effective way for replicating updates with Asynchronous Remote Stored Procedures.

Easy Centralized Administration

The Sybase Replication Server product includes, at no extra cost, a powerful systems management tool, the Replication Server Manager (RSM). The RSM graphical user interface allows administrators to manage from a single desktop not only individual replication system objects, but also collections of objects. RSM helps administrators check the status and monitor the performance of replication system components, including any heterogeneous data sources and targets. Managed system resources are represented as icons on the administrator's desktop. The object-oriented nature of the RSM interface shields administrators from network configuration details. Managed resources look the same regardless of their actual location.

The Replication Server Manager also includes monitoring status and performance of network connections, the space allocation and usage of queues, and the configuration of subscriptions.

High Availability of Data

Several features available with the Sybase replication product simplify the day-to-day operations of a replication system, making it easier for administrators to ensure high availability of corporate data at each business unit.

Automatic re-synchronization after failure; When a component is unavailable, data in the process of being delivered to a replicate site is stored in stable queues on disk. When the component returns to operation, the queued data resume their path to the replicate site. Replicate data is thus resynchronized automatically.

Coordination of database back-ups to facilitate system recovery; Backing up and restoring database information in a distributed replication environment can be challenging. To simplify the task, Sybase provides a mechanism for coordinating the back-up process across databases in a replication environment.

Bulk materialization from tape for site rebuild; When setting up replicate sites for the first time, or when rebuilding a replicate site it may not make sense to initialize the replicate data by transferring the entire database across the network. Replication Server allows administrators to export the primary data on to a tape and then load the data from the tape at the replicate sites. When the data from the tape is loaded at the replicate site, replication can resume without any loss of information.

Replication Server also supports Sun's high availability (HA) cluster configuration; A high availability cluster includes two machines that are configured so that, if one machine (or application) is brought down, the second machine automatically takes over. The cluster configuration tolerates these single-point failures:

- Server hardware failure
- Disk media failure
- Network interface failure
- Server OS failure

When any of these failures occur, the HA software fails over onto the other machine and restarts the Replication Server. To other components in the replication environment, it appears that the Replication Server experienced a reboot. The fact that it moved to another physical machine is transparent.

ASE offers an additional advantage for a replication environment: data mirroring. To maximize the availability of replication system components, and particularly sites that contain primary data, it may make sense to mirror the primary data on disks.

Heterogeneous Integration

Replication Server allows non-Sybase data sources to participate in a replication environment not only as targets for replicated data and procedures, but also as sources of such information.

Sybase Replication Server options offers the capability to capture changes from source databases, Oracle, DB2 UDB (on OS/390, NT and Unix), Microsoft SQL Server, or Informix, on a near real-time basis and send them to Replication Server, which delivers them to the target location. Replication Server can translate data as it is replicated between different databases. By defining the content and the format of datatypes that are unique to a heterogeneous environment, Replication Server is able to move data from one heterogeneous data source to another. For example, Replication Server can replicate a DB2 date column to an Informix data column. This feature reduces the need to create custom data transformation rules.

Replication of data and procedures to non-Sybase data sources is straightforward. Replication Server can replicate data to any data source accessible through Sybase DirectConnect, which includes DB2, UDB, Oracle, Informix, Microsoft SQL Server and any ODBC compliant database.

Local Autonomy

The Sybase Replication Server mechanism simply delivers data to a Sybase or non-Sybase data source. Consequently, each site retains full autonomy to decide how to view and how to use the replicated data. Each site is free to:

- Choose the subset (or full set) of primary data to be received and accessed.
- Set local names for data tables or columns.
- Optimize local data access methods.
- Modify data, either asynchronously with remote stored procedures, or synchronously with direct connections to the corresponding primary source of a piece of data.

Summary

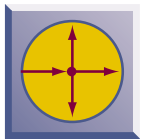
Sybase Replication Server, a key component of Sybase's Adaptive Server® Enterprise family of products, forms the foundation for data distribution in enterprise-wide client/server applications worldwide. Replication Server solves every one of the six requirements most important to businesses with distributed operations and distributed data:

- 1. High Availability of Data:** Data distribution through the reliable architecture of Replication Server reduces the exposure of business operations to computer system component failures.
- 2. Consistent Information Delivery:** Replication Server delivers information while maintaining the transactional integrity of the data.
- 3. High Performance:** Replication Server does not burden the original source of the data to be replicated, uses the network efficiently through intelligent routes, and allows each site to optimize local access to replicated data.
- 4. Easy Centralized Administration:** Replication Server includes a powerful systems management tool, the Replication Server Manager (RSM). From a single site, using a graphical user interface, RSM allows administrators to monitor and manage every component of the replication system.
- 5. Heterogeneous Data Source Access:** Replication Server Options allow customers to replicate data to and from non-Sybase data sources.
- 6. Local Autonomy:** Each site participating in a Sybase replication environment is free to decide which set of data it wishes to receive, how it will view the data, how it will access the data, and how it will modify the data.

Appendix: How Does One Set Up A Replication Server Environment?

Having described the role of each of the components of a Replication Server environment, let's now re-examine the examples that were introduced earlier in more detail. This section will illustrate possible ways of setting-up the replication examples.

Before we look in detail at the five replication examples introduced in the paper, let's discuss the building blocks (and their corresponding graphic icons) that will allow us to put together a replication environment.



Replication Server

This icon represents the Replication Server.



Adaptive Server Enterprise (ASE)

This icon represents the ASE. The data server manages databases that contain primary data, replicate data, or both.



Replication Agent

This icon represents the Sybase Replication Agent. The agent captures transactions from non-Sybase data servers and sends them to a Replication Server.



Non-Sybase Data Server

This icon represents non-Sybase data servers such as DB2, Oracle, or Microsoft SQL Server.



Replication Server Manager

This icon represents the Sybase Replication Server Manager (both the Replication Server plug-in for Sybase Central, and the RSM Server).

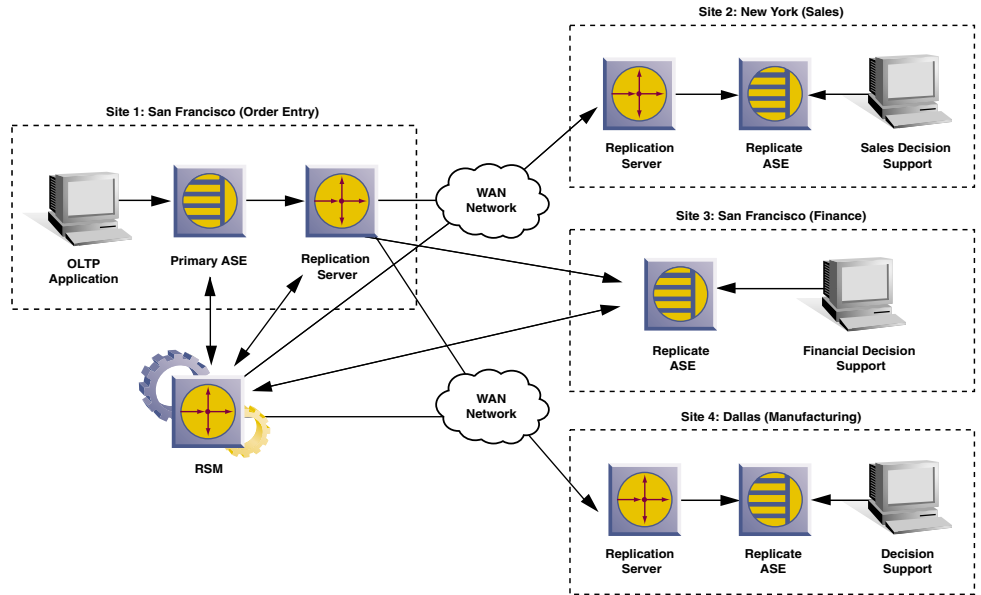


Client Application

This icon represents a client application. A client application is a user process or application connected to a data server. It may be a front-end application program executed by a user, or a utility program that executes as an extension of the system.

Example 1 Revisited: Decision Support Replicates

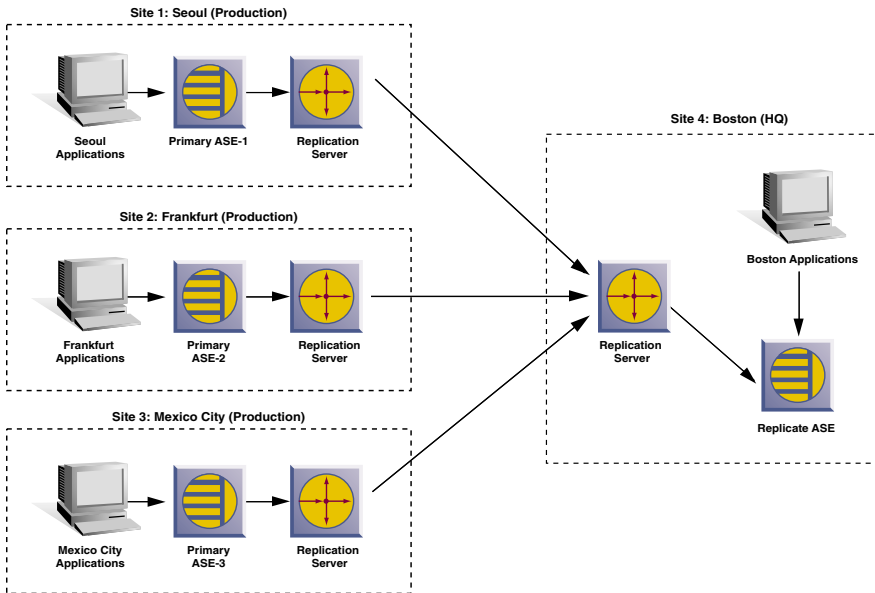
Figure 3 of the paper provided an illustration of a simple replication environment where a subset of the order entry data in San Francisco is replicated for decision support purposes to several other business units, including Finance (also in San Francisco), Sales (New York) and Manufacturing (Dallas). Figure 14 illustrates a possible configuration for this scenario. Note that two direct wide area network routes have been set up which connect the Replication Server in San Francisco to the Replication Servers in New York and Dallas. The replicate database in Finance, which happens to be on the same LAN as the primary data server does not require a separate Replication Server.



[Figure 14.] Decision Support Replication (Components used in configuration Example 1)

Example 2 Revisited: Corporate Data Consolidation

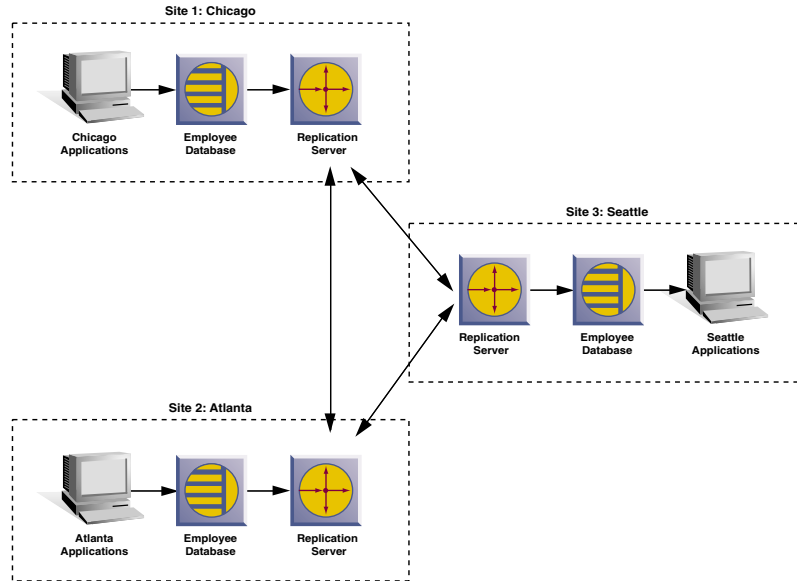
The graphic below provides details on a possible configuration for the replication example illustrated in Figure 5 of the paper. In this case distinct data items are replicated from three distributed sites (Seoul, Frankfurt, and Mexico City) to a consolidated database in Boston.



[Figure 15.] Corporate Data Consolidation (Components used in configuration example 3)

Example 3 Revisited: Corporate Data Sharing

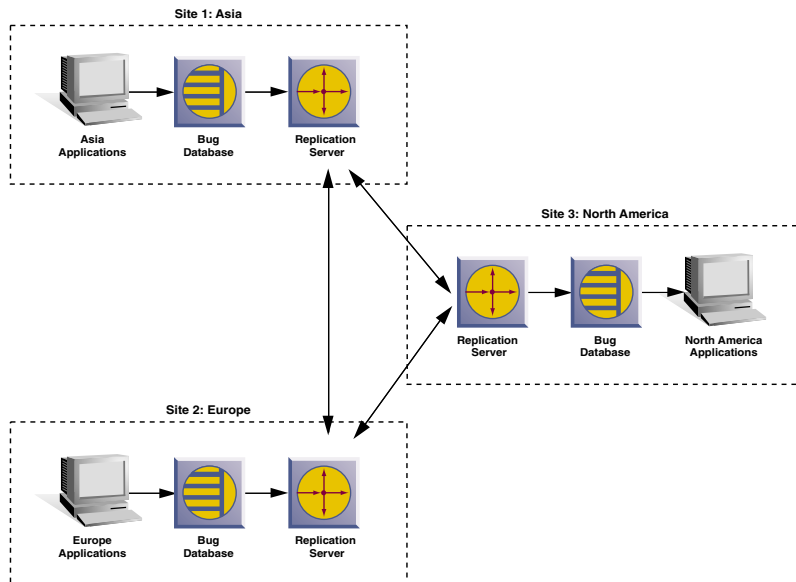
The graphic below provides details on a possible configuration for the replication example illustrated in Figure 6 of this paper. In this case, distinct data items (local employee information) are being replicated from each of three distributed sites (Chicago, Atlanta, Seattle) to all other sites. As a result, all three sites can share information that originates in all three locations.



[Figure 16.] Corporate Data Sharing (Components used in configuration example 4)

Example 4 Revisited: Corporate Data Integration

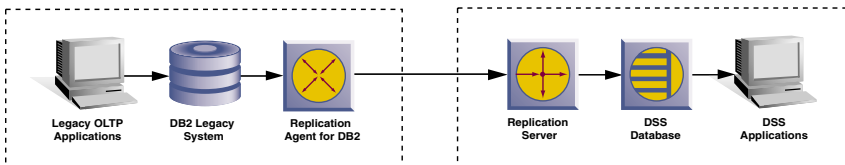
The graphic below provides details on a possible configuration for the replication example illustrated in Figure 7 of the paper. Customer support information entered by support analysts in each site is being replicated from each of three distributed sites. As a result, all three sites can share information that originates in all three locations. Unlike the previous case, the customer application in this case will need to include a mechanism for resolving conflicts. As discussed in the paper, Sybase recommends a configuration where at any point in time each data item has a clearly identifiable site that owns it.



[Figure 17.] Corporate Data Intergration (Components used in configuration example 3)

Example 5 Revisited: Non-Sybase Data Replication

The graphic below provides details on a configuration for the replication example illustrated in Example 7 of the paper. The Sybase Replication Agent for DB2 is used to capture transactions in DB2 on an IBM mainframe. The transactions are replicated to an ASE that is used for decision support by management.



[Figure 18.] Non-Sybase Data Replication

International Contacts

Argentina +5411 4313 4488	Korea +82 2 3451 5200
Australia +612 9936 8800	Malaysia +603 2142 4218
Austria +43 1 504 8510	Mexico +5255 5093 8500
Belgium +32 2 713 15 03	Netherlands +31 20 346 9290
Brazil +5511 3046 7388	New Zealand +64 4473 3661
Bulgaria +359 2 986 1287	Nigeria +234 12 62 5120
Canada +905 273 8500	Norway +47 231 621 45
Central America +506 204 7151	Panama +507 263 4349
Chile +56 2 330 6700	Peru +511 221 4190
China +8610 6856 8488	Philippines +632 750 2550
Colombia +57 1 218 8266	Poland +48 22 844 55 55
Croatia +385 42 33 1812	Portugal +351 21 424 6710
Czech Republic +420 2 24 31 08 08	Puerto Rico +787 289 7895
Denmark +45 3927 7913	Romania +40 1 231 08 70
Ecuador +59 322 508 593	Russian Federation +7 095 797 4774
El Salvador +503 245 1128	Slovak Republic +421 26 478 2281
Finland +358 9 7250 200	Slovenia +385 42 33 1812
France +33 1 41 91 96 80	South Africa +27 11 804 3740
Germany +49 69 9508 6182	South Korea +82 2 3451 5200
Greece +30 1 98 89 300	Spain +34 91 749 7605
Guatemala +502 366 4348	Sweden +46 8 568 512 00
Honduras +504 239 5483	Switzerland +41 1 800 9220
Hong Kong +852 2506 6000	Taiwan +886 2 2715 6000
Hungary +36 1 248 2919	Thailand +662 618 8638
India +91 22 655 0258	Turkey +90 212 325 4114
Indonesia +62 21 526 7690	Ukraine +380 44 227 3230
Israel +972 3 548 3555	United Arab Emirates +971 2 627 5911
Italy +39 02 696 820 64	United Kingdom +44 870 240 2255
Ivory Coast +225 22 43 73 73	Venezuela +58 212 267 5670
Japan +81 3 5210 6000	Asian Solutions Center +852 2506 8700
Kazakstan +7 3272 64 1566	

For other Europe, Middle East, or Africa inquiries:
+33 1 41 90 41 64 (Sybase Europe)

For other Asia Pacific inquiries:
+852 2506 8700 (Hong Kong)

For other Latin America inquiries:
+925 236 6820



Sybase, Inc. Worldwide Headquarters

One Sybase Drive
Dublin, CA 94568-7902 USA
Tel: +800 8 SYBASE
www.sybase.com

Copyright © 2003 Sybase, Inc. All rights reserved. Unpublished rights reserved under U.S. copyright laws. Sybase, the Sybase logo, Power Builder and Adaptive Server are trademarks or registered trademarks of Sybase, Inc. or its subsidiaries. All other trademarks are the property of their respective owners. ® indicates registration in the United States of America.

This Technical White Paper was prepared by Sybase, Inc. L02038 MIL6143