



**Enterprise replication:  
a high-performance solution for  
distributing and sharing information**





---

## Table of Contents

---

4	<b>Introduction</b>
5	<i>Enterprise replication: an overview</i>
7	<i>What is data replication?</i>
8	<i>Data replication: selecting the type that is right for you</i>
8	<i>Master/slave ownership</i>
9	<i>Data dissemination ensures central data control</i>
10	<i>Data consolidation lets data be updated at the source</i>
11	<i>Workload partitioning lets data be controlled locally yet viewed globally</i>
12	<i>Workflow ownership makes ownership move</i>
13	<i>Update-anywhere replication allows local updates while ensuring consistent data enterprisewide</i>
14	<i>Change capture: trigger-based versus log-based</i>
17	<i>IBM IDS enterprise replication: an architecture with advantages</i>
17	<i>High performance</i>
18	<i>High availability</i>
19	<i>Consistent and reliable transactional integrity</i>
19	<i>Flexibility</i>
20	<i>Manageability</i>
21	<i>IBM IDS enterprise replication: a technical overview</i>
22	<i>Configuration and control</i>
23	<i>Replication network topologies</i>
26	<i>Transaction capture</i>
27	<i>Transaction queuing and delivery</i>
28	<i>Transaction insertion at target server</i>
28	<i>Conflict detection and resolution</i>
29	<i>What's new for existing enterprise replication users?</i>
30	<i>Why IBM IDS enterprise replication?</i>



### Introduction

Over the last decade, corporate computing has changed the role of the mainframe. No longer is it the sole corporate information resource. This migration continues to be driven by competitive, cost-cutting and organizational forces, which necessitate that companies maximize their IT resources. One result of this decentralization trend is that IS managers find themselves managing distributed databases that spread corporate information to all corners of the enterprise.

Data replication is key to effectively distributing and sharing this information. But replication involves more than merely moving data around from site to site. It is an extremely demanding process. And as such, the success of replication systems depends on technology that addresses the full range of business and application needs.

Replication systems must provide high performance without burdening the source database and without requiring application changes. They have to maximize system availability and ensure consistent delivery of the data. And, finally, database administrators need to be able to configure and manage all the components of the replication system in a manner that utilizes their enterprise computing resources most effectively. This paper explores these requirements and how IBM Informix® Dynamic Server™ (IBM IDS) helps enterprises ensure successful deployment of this important capability. We begin with an overview of IBM IDS technology followed by some key replication concepts and considerations. We then examine the IBM IDS replication solution—enterprise replication (ER)—in more detail.



**Enterprise replication: an overview**

IBM IDS ER, a high-performance replication solution, provides an enterprisewide foundation for distributing and sharing corporate information. Enterprise replication supports replication needs from the workgroup to high-end SMP systems. A number of key factors characterize the distinct advantages of this solution, which meets a wide spectrum of business and application requirements:

- **High performance.** The industry-leading IBM IDS technology provides superior performance, scalability and manageability for high-volume, enterprisewide applications. IBM IDS offers the first RDBMS replication solution that takes full advantage of scalable parallel-processing server architectures to meet the most demanding computing environments.
- **High availability.** IBM IDS replication products assure high availability through a variety of options—everything from hot standby servers to asynchronous replication of data to single or multiple sites. With asynchronous replication, network and target node outages are tolerated, minimizing the impact of a failure at any point. Updates to be replicated, including changes to the IBM IDS ER global catalog, are automatically propagated to the remote sites when replication processing is resumed.



- **Transaction-level integrity.** Consistency is maintained by immediately propagating database changes from the source database to the target database. Transactions are replicated so that transactional ordering is preserved. IBM IDS ER also provides a number of options for detecting and resolving update conflicts.
- **Flexible architecture.** IBM IDS ER provides a replication architecture that allows organizations to define their replication environments based on specific business and application requirements. IBM IDS ER supports a variety of replication usage models, including master/slave, workflow updates and update-anywhere. Both fully connected and hierarchical replication network topologies are supported. IBM IDS ER implementation is transparent to applications, thereby avoiding code changes.
- **Manageability.** Configuration and monitoring can be performed through a set of browser-based tools that are integrated with the database server management framework. A command line interface is also provided to facilitate the use of scripts.

Before describing IBM IDS ER in detail, we will outline some key replication concepts and considerations for organizations planning to incorporate replication technology.



### **What is data replication?**

Replication can be simply defined as the process of generating and reproducing multiple copies of data at one or more sites. Database replication is important because it enables enterprises to provide users with access to current data where and when they need it. Data replication can provide a wide spectrum of benefits, including improved performance when centralized resources get overloaded, increased data availability, capacity relief and support for data warehousing that facilitates decision support.

Replication is either synchronous or asynchronous. Each type offers capabilities and strengths suited for particular purposes. In synchronous replication, the replicated data is updated immediately whenever the source data is updated. During synchronous replication, data integrity is typically protected with a two-phase commit protocol. After each server updates its data, it sends a confirmation message to the source database and waits for the response command before completing the transaction. While synchronous replication is a good technique for applications that demand immediate data synchronization (such as financial transactions), two-phase commit has high overhead and can significantly degrade performance. In addition, the transaction monitor used to implement this approach generally introduces a single point of failure into the system, thus impacting availability.

With asynchronous replication, on the other hand, the target database is updated only after the source database has been modified. This delay in updating can range from a few seconds to several hours, depending on the configuration, with the data eventually synchronized to the same value at all sites. If a particular site fails or is not accessible, asynchronous replication allows requests for local processing to continue. For example, in an order-entry system, orders can continue to be entered even if the shipping site goes down. When the failed site comes back online, the system guarantees propagation of those entries.



**Data replication: selecting the type that is right for you**

Since replication plays a major role in enabling distributed computing, organizations need to understand its uses and plan how best to deploy the technology. A fundamental issue is the trade-off between data integrity and availability. As we have described, synchronous technology ensures integrity while sacrificing availability and response time performance. At the other end of the replication spectrum, asynchronous technology maximizes availability and response time performance, although planning and design are required to ensure data integrity and resolve update conflicts. Ultimately, the appropriate deployment strategy is determined by business and application requirements. The replication system needs to support these requirements. Another fundamental issue in asynchronous replication is data ownership. Ownership refers to the authority of a site to update data. The type of ownership determines the need, if any, for conflict detection and resolution. For example, in a master/slave configuration, only one site may update data, usually the master unless otherwise specified, eliminating any need for conflict detection and resolution. Other ownership models, as discussed later, require the use of conflict detection and resolution since they allow information to be updated at multiple sites.

**Master/slave ownership**

Master/slave ownership allows several usage scenarios, including data dissemination, data consolidation and workload partitioning.

**Data dissemination ensures central data control**

Data dissemination involves the updating of data at a central location and its subsequent replication to regional read-only sites. This method is useful when information must be distributed to branch locations. For example, a book store chain headquarters may need to send updated price lists of available books to its stores on a nightly basis. To ensure that this data is consistent, the stores have read-only access to the information while the headquarters has read-and-write capabilities.

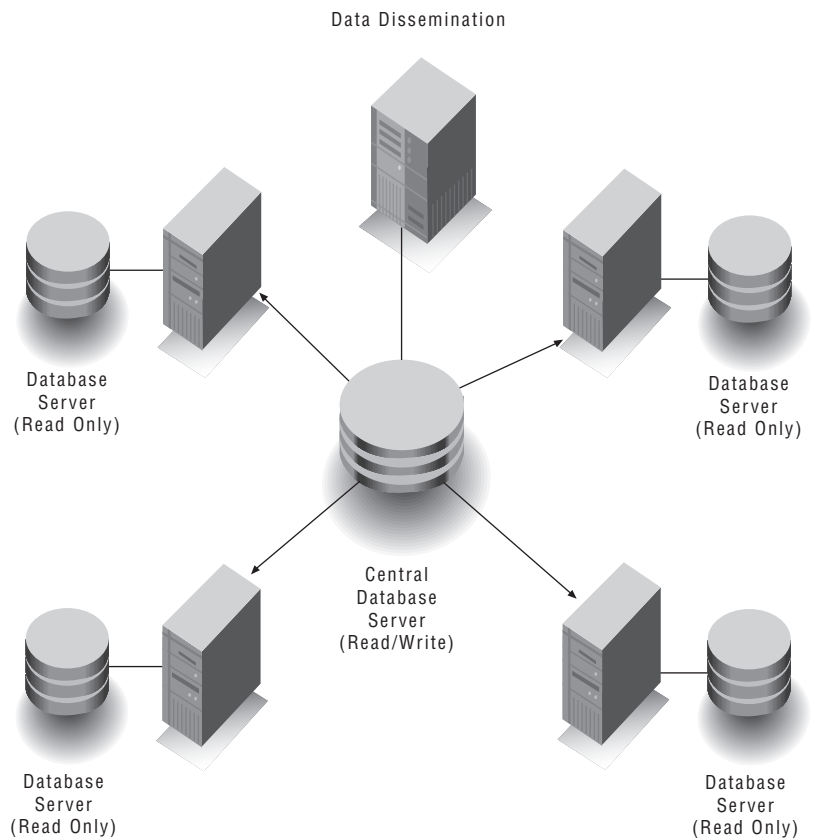
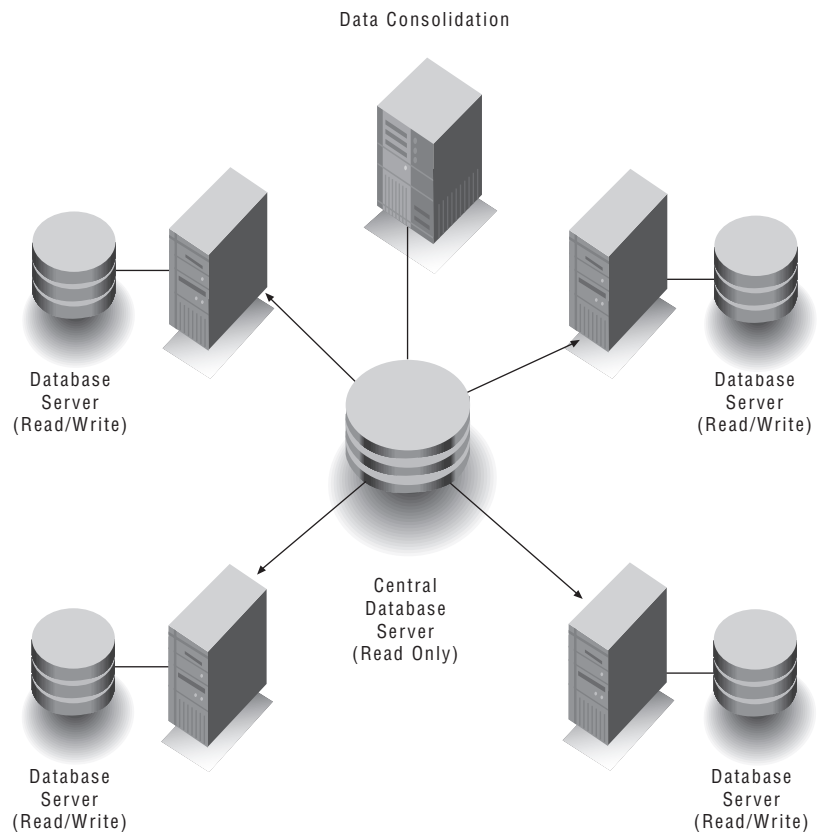


Figure 1. With data dissemination, data is updated at a central location and replicated to multiple, read-only sites.



**Data consolidation lets data be updated at the source**

Data consolidation involves regionally updating data sets, which are then brought together in a read-only repository on the central database server. This method gives data ownership and location autonomy at the branch level. An example of such a data consolidation environment is a retail store chain that throughout the day gathers point-of-sale information. At the end of the business day, the stores must transmit the data to headquarters, where it is consolidated into the central data warehouse to be used in various business intelligence processes, such as trend analysis.



*Figure 2. With data consolidation, data is updated at multiple sites and replicated to a central, read-only site.*



**Workload partitioning lets data be controlled locally yet viewed globally**

Workload partitioning gives database administrators (DBAs) the flexibility to assign ownership of data at the table partition level. An example is illustrated below. The replication schema is mapped to the partitioning schema for the employee tables. The Asia/Pacific site has ownership of its partition and can update, insert and delete employee records for personnel in its region. Any changes to the data are then propagated to the U.S. and European sites. While the Asia/Pacific site can query or read the other partitions, it cannot update them. Similarly, the U.S. and European sites can change data only within their own respective partitions but can query and read data in all partitions.

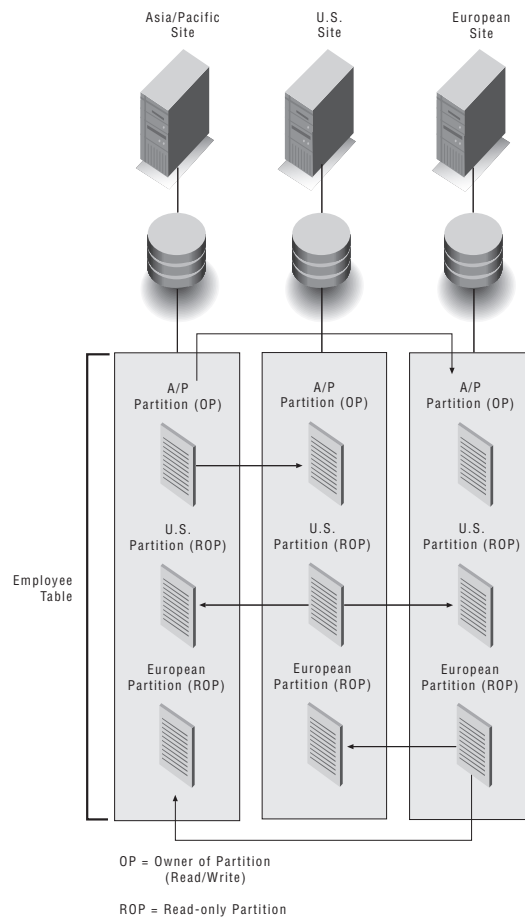
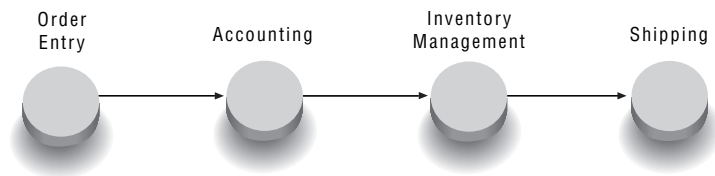


Figure 3. With workload partitioning, users at different sites can update data only in their own partition but can view data in the entire table.

**Workflow ownership makes ownership move**

As in master/slave ownership, workflow ownership avoids IBM IDS update conflicts, but it provides a more dynamic update technique than master/slave affords. Workflow ownership enables data to advance from site to site. Only one site, however, may update the data at any given instant. Each site is dependent on the data from the previous sites, and each can update only data that is relevant to its business function. At the completion of each step, data is updated and replicated to the next site.

A typical workflow example is an order-processing system. An order is first entered by the order-entry department. The order is then sent to the accounting department for credit approval and invoicing, and then on to inventory management for authorization. Finally, the order is routed to the shipping department for packing and delivering.



*Figure 4. In a typical workflow example, the authority to modify data advances from site to site.*

**Update-anywhere replication allows local updates while ensuring consistent data enterprisewide**

Organizations also require the ability to support realtime updates at multiple sites. Unlike master/slave ownership, in which replicated data is read-only, update-anywhere replication occurs in a peer-to-peer environment in which multiple sites have equal authority to update data. This allows local users to function autonomously, even when other systems or networks in the distributed environment are not available. To resolve update conflicts, however, the replication architecture must support this environment with a wide range of automatic conflict detection and resolution routines. An example of update-anywhere is illustrated in the figure below. In this example, a mail-order company provides a toll-free number for placing orders. A call gets routed to one of three call centers, depending on the time of day and call load-balancing requirements. Each call center consequently needs the ability to update records and have them replicated to the other call centers as soon as orders are placed.



*Figure 5. In an update-anywhere scenario, data may need to be owned and updated at some sites, while other sites have read-only copies.*



**Change capture: trigger-based versus log-based**

The most frequently used replication change capture mechanisms are trigger-based and log-based capture. In trigger-based capture, modifications to the data set off triggers inside the database. This activates replication-specific code inside the source database which, in turn, initiates the replication process. This process occurs simultaneously within the same unit of work as the user transaction, and therefore directly impacts the performance of transaction processing at the source database.

Another core problem with trigger-based systems stems from how trigger-based systems evolved from generic database triggers. Database triggers were initially introduced by database vendors to protect the referential integrity of corporate data and to centrally enforce business rules. For these vendors, trigger-based replication was a logical and straightforward extension to their products. Also, trigger-based replication provides the flexibility for third-party vendors to implement heterogeneous replication, since most DBMS vendors will not disclose their internal log formats. A consequence, however, is that trigger-based replication is handicapped because vendors had to introduce additional functionality to ensure transactional integrity, and this added functionality imposes a severe performance and administrative burden.

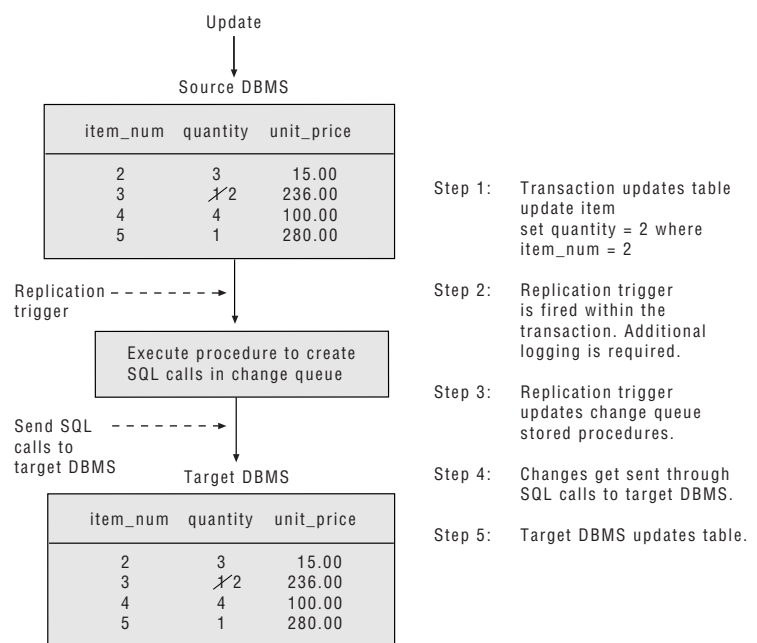
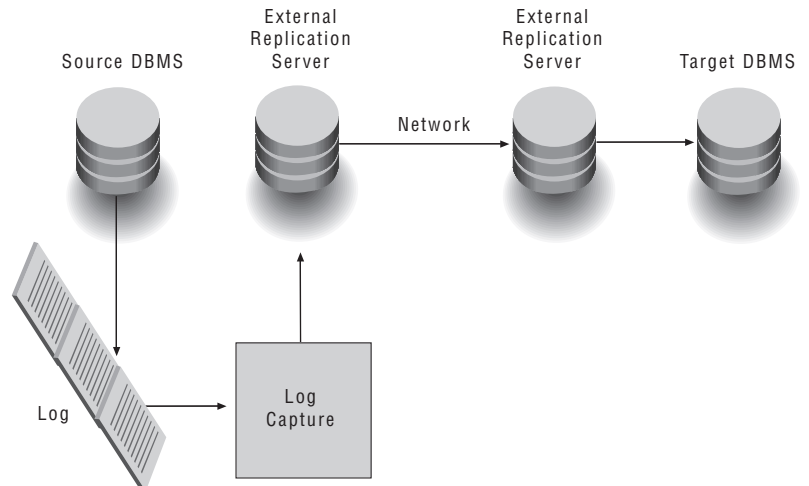


Figure 6. Trigger-based replication adversely impacts the performance of the original transaction.

Log-based systems, on the other hand, can operate as part of the normal database logging process, with minimal overhead added to the system. The transaction log is viewed as the best source for capturing changes to the source data. Modifications to the source database are detected and propagated without interfering with the normal operations of the source database system.

However, all log-based replication systems are not the same. One key differentiator is the degree of integration of the log capture mechanism with the database system. For example, log-based products that utilize external servers and processes offset any advantages of log-based transaction capture, owing to the additional data transfer overhead. Another characteristic affecting the success of log-based systems involves whether every entry in the log must be read as part of the transaction capture process.



*Figure 7. External replication servers used for log-based capture incur additional data transfer overhead.*

The weaknesses inherent in trigger-based architectures and poor log-based implementations were examined prior to designing the IBM IDS ER solution. As a result, IBM IDS ER is developed around a high-performance, log-based transaction capture and distribution mechanism which is an integral part of the database system.

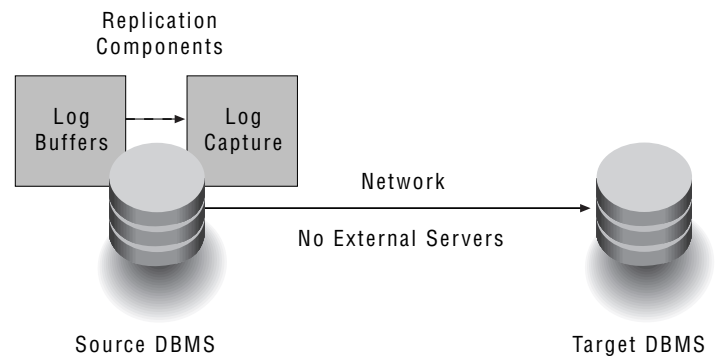


Figure 8. Replication integrated with the DBMS eliminates the need for external servers and minimizes log capture overhead.



**IBM IDS enterprise replication: an architecture with advantages**

The introduction of IBM IDS ER brings to the market a highly advanced replication solution. Built around the industry-leading Dynamic Scalable Architecture (DSA), IBM IDS ER delivers best-of-class technology while avoiding the limitations of other replication alternatives. IBM IDS ER supports systems ranging from workgroup servers to high-end SMP systems and was designed to address a full range of business requirements including:

- High performance
- High availability
- Consistent and reliable transactional integrity
- Flexible architecture
- Manageability
- Heterogeneous data source support.

**High performance**

DSA provides an optimal high-performance, scalable platform for supporting the demands of enterprisewide data replication systems. Early on, the limitations of trigger-based replication were recognized, and it was decided to implement a highly efficient log-based transaction capture and parallel distribution mechanism integrated with the database architecture. This approach has been implemented without the performance degradation and administrative burden inherent with other log-based systems that utilize external servers.

In addition, DSA implements a “parallel everything” approach by exploiting the inherent parallel processing power of SMP architecture when such servers are used. The result is that all replication operations are performed in parallel, further enhancing the performance of the system and minimizing bottlenecks.



**High availability**

To maximize availability and fault tolerance, IBM IDS ER allows for a central database to be replicated to a single secondary server. This option is useful for creating a hot standby server to take over processing if the primary server fails.

IBM IDS ER also protects against primary system failures through asynchronous replication of data to one or multiple secondary sites. Any updates at the primary site, including changes to the IBM IDS ER global catalog, are automatically propagated to the secondary sites, ensuring that all sites have consistent replicas of the data. Transmission of the updates can be immediate or time-driven, in which case the DBA specifies time intervals for the updates. Updates can also be event-driven, such as after a transaction commit or as specified by the user.

IBM IDS ER also utilizes a reliable message delivery mechanism, which stores data locally and propagates the data to the remote server as a separate transaction. In the event of a server or network failure, the local server can continue to service local users, providing a high degree of fault tolerance. After the failed server or network has resumed operation, all changes to the source database are propagated to the remote databases.



#### **Consistent and reliable transactional integrity**

IBM IDS ER maintains multiple, consistent copies of data at different sites in a distributed environment. This is accomplished by asynchronously propagating database changes resulting from a transaction to the target server immediately after they are committed at the source server. In addition, transactions are stored to maintain transactional consistency and ordering. This ensures that in the event the same record is updated multiple times, the changes will be applied to the remote databases in the same order as they were applied to the primary database.

To prevent update conflicts, IBM IDS ER provides built-in, automatic conflict detection and resolution through a variety of predefined routines or, alternatively, through customized resolution methods.

#### **Flexibility**

IBM IDS ER meets a wide spectrum of business and application requirements. First, it supports the full range of ownership models—master/slave, workflow and update-anywhere—using fully connected or hierarchical replication network topologies.

Second, IBM IDS ER provides support for replication of subsets and partitions of tables. IBM IDS ER also allows multiple schemas for replicated servers, providing DBAs with complete discretion in defining the section of the database to be replicated, even down to the row and column level. This flexibility enables DBAs to customize the database for specific business needs. For example, operational data for transaction-oriented databases can be organized differently from data for data warehouses, which is designed to accommodate queries and analysis.



### **Manageability**

In developing IBM IDS ER, significant investment was made in both replication technology and, of equal importance, manageability. DBAs can perform replication administration tasks by using the IBM Informix Server Administrator (ISA), a cross-platform, browser-based tool. Commands can also be submitted with the `cdr` command line utility, which facilitates the use of dynamically generated scripts. Through administration commands, the DBA can select the types of replication methods, designate the target servers, define read/write capabilities and specify replication frequency. Furthermore, unlike other replication products, IBM IDS ER allows DBAs to dynamically change the replication configuration, enabling them to add and configure new servers online without shutting down the entire system. Servers can be connected and disconnected without disrupting the replication process, since transactions are queued until the connection is reestablished. This feature allows the DBA to set up the environment to send data only when the network is idle or communication costs are lower and to balance the load on different servers at different times of day.

Another key feature is the global replication catalog, which keeps track of IBM IDS ER configuration information. It maintains information such as replication definitions, IBM IDS ER servers, subscriber lists, default queuing thresholds and conflict detection and resolution rules and their associated stored procedures. A major benefit of the global replication catalog to DBAs is that it automatically propagates any changes to the replication definition to the other servers on the network. This relieves the DBA of the manual task of copying the data to remote servers, which, in turn, minimizes potential inconsistencies within the global catalog. Monitoring capabilities include the ability to monitor the size of replication queues to each destination, view the availability of remote servers and make immediate notification of conflicts among the replication servers.



**IBM IDS enterprise replication: a technical overview**

IBM IDS ER includes the following, discussed in detail in this section:

- Configuration and control
- Replication network topologies
- Transaction capture
- Transaction queuing and delivery
- Transaction insertion at target server
- Conflict detection and resolution.

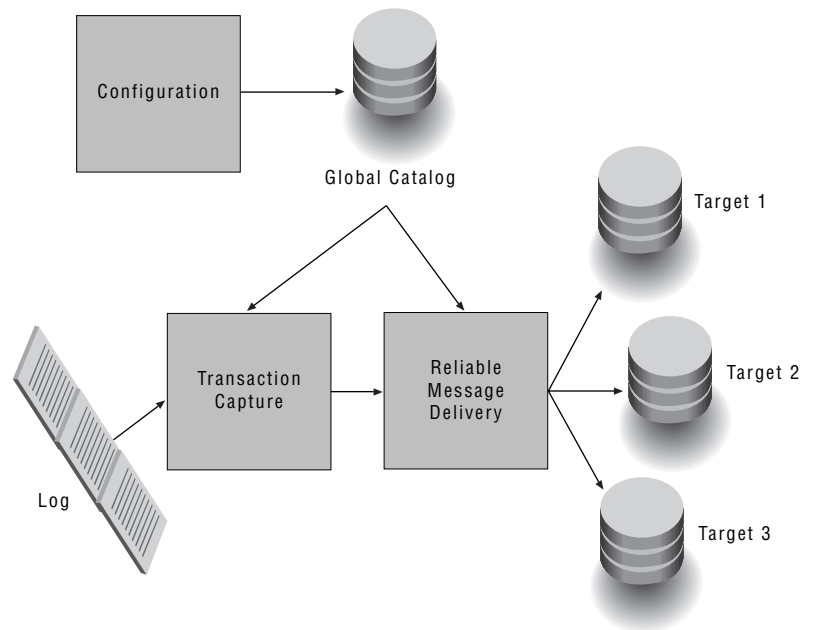


Figure 9. IBM IDS enterprise replication components.



**Configuration and control**

IBM IDS ER provides a flexible method for configuring replication definitions that ensures consistent transaction capture and delivery. Replication can be defined at the table level or by using table subsets such as partitions or views. The replication group feature allows replicated definitions to be grouped together. Replication groups simplify administration and are easy to manage, since all replication commands at the table level are applicable at the group level. This feature also allows IBM IDS ER to speed up processing at the target databases by applying in parallel transactions that participate in different replication groups.

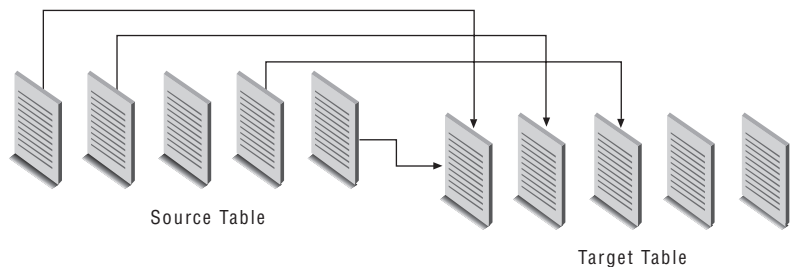


Figure 10. Replication can be defined to include table subsets.

### Replication network topologies

Enterprise replication topology describes the connections that replication servers make to interact with each other. IBM IDS ER supports two types of network topology:

- Fully connected
- Hierarchical replication.

A fully connected topology is one in which all database servers connect to each other, with IBM IDS ER managing the connections. In a fully connected topology, replication messages are sent directly from one server to another, with no additional routing required.

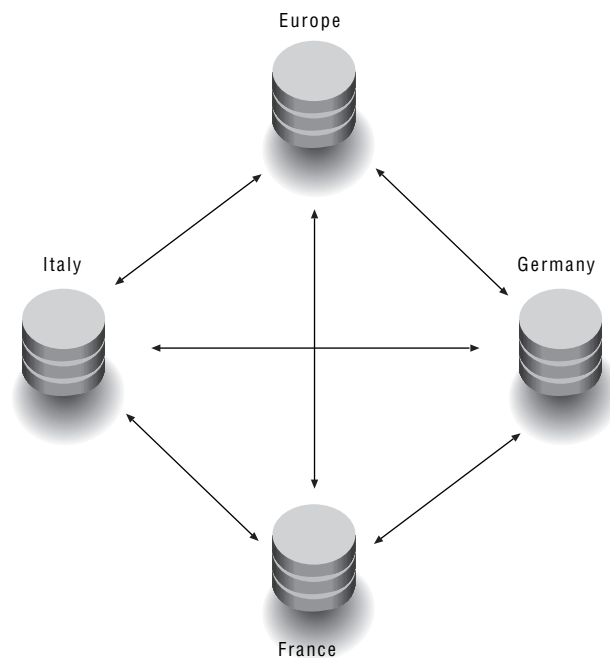


Figure 11. Fully connected topology.

The two types of hierarchical routing topologies are:

- Hierarchical tree
- Forest of trees.

A hierarchical tree consists of a root database server and one or more databases organized into a tree. Each server in the tree references its parent except for the root server, which has no parent.

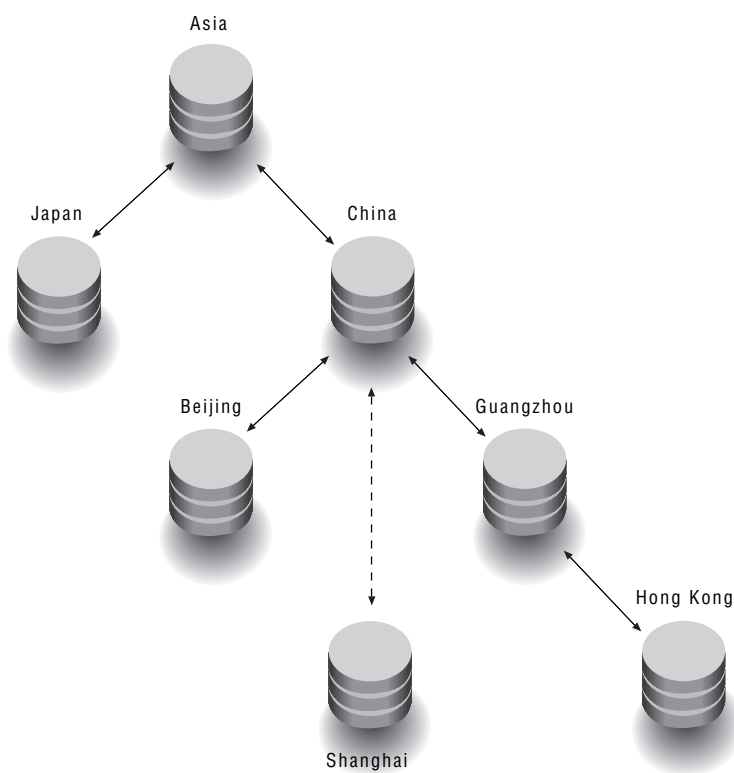


Figure 12. Hierarchical tree topology.

A forest-of-trees topology is one in which several hierarchical trees have root database servers that are fully connected. In Figure 13, North America, Europe and Asia are root database servers and they transfer replication messages to each other for delivery to their child nodes.

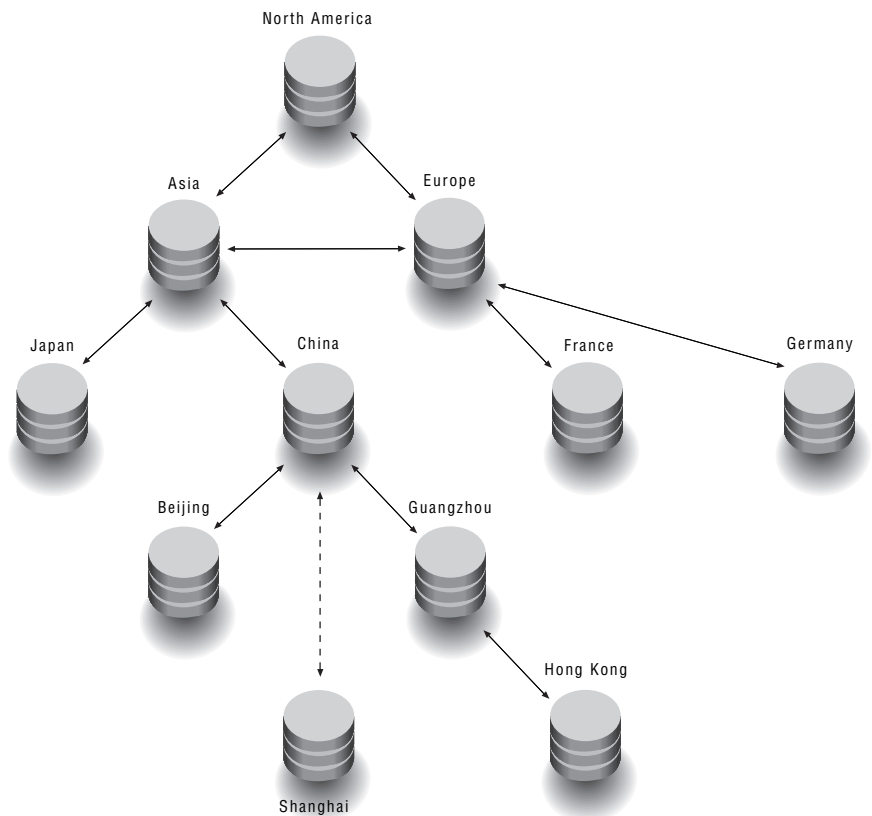


Figure 13. Forest-of-trees topology.

Organizing the database servers in a hierarchical tree or a forest-of-trees greatly reduces the number of physical connections required to make a replication system.

### Transaction capture

To capture replicated transactions, IBM IDS ER employs a transaction log post-processing method. Post-processing provides a unique performance advantage by minimizing impact on transaction processing. First, IBM IDS ER reads the log buffers only for the records that are earmarked for replication as opposed to the entire log. The participating records are then filtered to determine if they meet any of the replication definitions. Qualifying records are collected into a transaction group. If a transaction updates the same row multiple times, only the last version is collected. To assure high performance, IBM IDS ER uses parallel processing to filter the records. Also, no additional I/O is utilized since this process is integrated with the database engine.

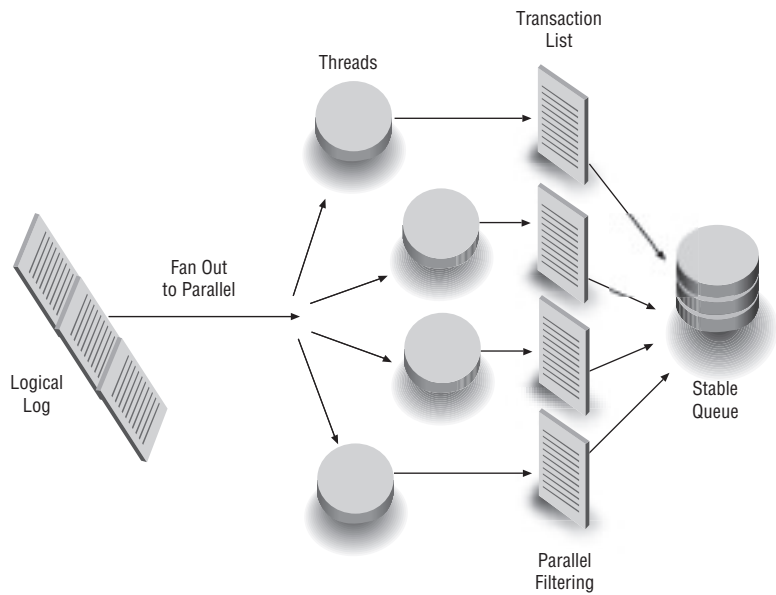


Figure 14. Enterprise replication uses parallel processing to filter the records.



**Transaction queuing and delivery**

When a replicated transaction actually commits, the transaction is placed in a “stable” queue for delivery, and aborted transactions are discarded. A DBA can choose whether or not to compress the data that is sent over the network by using standard data compression algorithms. Doing so can improve performance over slow links if the data compresses well (such as text). In the event of a network failure, the stable queue holds the transactions until the network is restored. To ensure integrity, the commit order is maintained during delivery. Once the target site receives the changes, an acknowledgment is sent to the source server to confirm delivery.

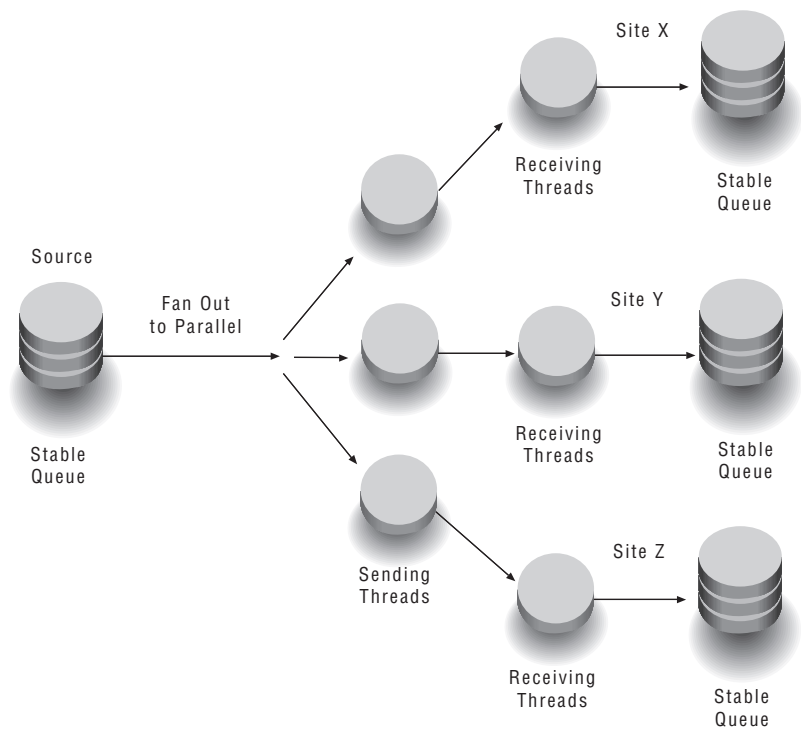


Figure 15. Queuing and delivery of a transaction.

### Transaction insertion at target server

The transaction groups are moved from the stable queue and inserted into the appropriate tables as database transactions. This is illustrated in Figure 16. To improve performance, this operation supports parallel insertion options. If a replicated record has the same primary key as an existing record in a target table, the preconfigured update detection and resolution rule is applied to determine which record prevails. All of the necessary configuration data is stored in the global catalog.

### Conflict detection and resolution

Enterprise replication provides integrated update conflict detection and resolution facilities that invoke application-specific routines to restore data consistency. When an update conflict is detected, the following routines can be applied on a table-by-table basis to determine record priority:

- **Latest timestamp.** Gives the record with the latest timestamp priority.
- **Stored procedures.** Allows the DBA to develop application-specific procedures for conflict resolution using IBM Informix Stored Procedure Language (SPL). For example, DBAs can designate specific servers to have higher priorities than other servers. Transactions that are not processed can be cached for subsequent investigation and reconciliation.
- **Ignore.** Used to disable conflict detection and resolution, which is useful in master/slave configurations.

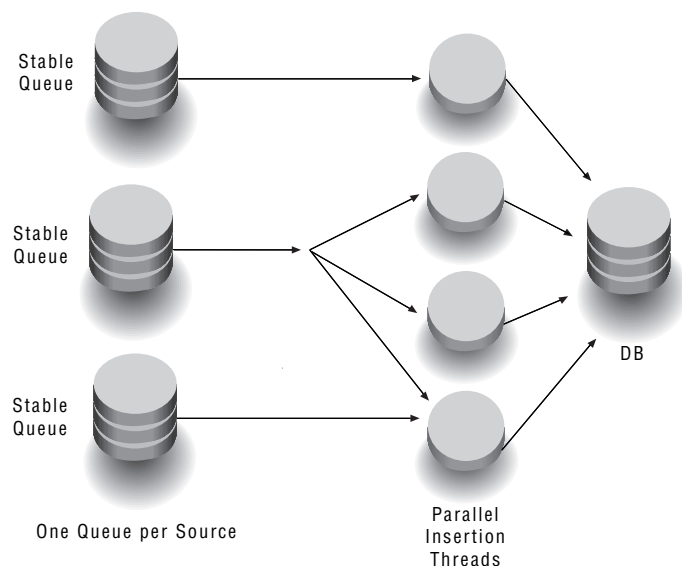


Figure 16. Transaction groups are moved from the stable queue and inserted into the appropriate tables as database transactions.



**What's new for existing enterprise replication users?**

Many enhancements have been made to IBM IDS ER since it was first introduced. They are summarized below:

- A comprehensive cdr utility, which allows the DBA to configure and control enterprise replication from the command line in a Microsoft® Windows® or UNIX® system.
- Connection and disconnection of servers without disrupting the enterprise replication process. Transactions will be queued until the connection is reestablished. This can be used to balance the load on database servers and the network.
- Compression of network messages using standard data compression algorithms. For messages that compress well, such as text, this can result in performance improvement in slow networking environments.
- Hierarchical routing topologies that can significantly reduce the number of server-to-server connections in a large network.
- Compression of transactions by including only the last change to a row.
- Support for replication of user-defined opaque types and smart large objects, including spatial objects.
- Improved parallelism when applying transactions.
- Support for serial primary keys.
- An in-place ALTER TABLE that allows replication to be defined on an existing table without having to rebuild it.
- Improved stable queue spooling, which improves performance and reduces the probability of blocking states.



**Why IBM IDS enterprise replication?**

In today's world of increasingly dispersed data, distributed database capabilities have become a necessity rather than a luxury. Replication plays a crucial role in enabling distributed database systems, particularly as organizations move to more downsized, decentralized business models. Organizations considering replication need to plan for how best to deploy the technology to meet their needs. Replication benefits include improved performance for computing resources, increased data availability, capacity relief and support for data warehousing that facilitates decision support. IBM IDS ER delivers high performance, availability, consistent data distribution and manageability—all the elements necessary to meet the demands of distributing and sharing information throughout the enterprise.



**The first step to enterprise replication**

To get started or receive more information, call your local IBM Data Management sales representative or 1 800 331 1763. Only IBM Data Management has the experience, technology and resources to help your organization take full advantage of replication technology and make your distributed computing solution a success. For more information on IBM Informix Dynamic Server, or any other IBM Data Management products or services, please contact your local sales office.

**Find out more**

For more information, contact the nearest sales office or visit the Web site at [ibm.com/software/data/informix](http://ibm.com/software/data/informix)



© Copyright IBM Corporation 2001

IBM Corporation  
Silicon Valley Laboratory  
555 Bailey Avenue  
San Jose, CA 95141  
U.S.A.

Printed in the United States of America  
12-01  
All Rights Reserved

Dynamic Server, IBM, the IBM logo and Informix are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries or both.

Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States, other countries or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product and service names may be trademarks or registered trademarks of their respective companies.

The information in this paper is provided by IBM on an "as is" basis without any warranty, guarantee or assurance of any kind. IBM also does not provide any warranty, guarantee or assurance that the information in this paper is free from any errors or omissions. IBM undertakes no responsibility to update any information contained in this paper.



Printed in the United States on recycled paper containing 10% recovered post-consumer fiber.



GC27-1570-00