

Oracle7

Distributed Database Technology and Symmetric Replication

APRIL 1995

Oracle7

Distributed Database Technology and Symmetric Replication

Distributed database technology is a fundamental component of a distributed computing solution.

Introduction

Your enterprise probably doesn't operate from a single room, a single building, or even a single continent. Most likely, you have users generating and demanding access to data from points both near and far. You have users working in many separate functional units across your enterprise, users that need computing systems that enable them to work together effectively wherever they might be.

Linking these operations together requires distributed computing systems that allow data to be shared in many different ways. These systems need to support client GUI front-end processing, application processing, and database processing where it makes the most sense to maximize performance and availability. Most of all, distributed computing systems need to be manageable.

Oracle offers the most advanced technologies for distributed computing, including:

- Client/server computing over almost any network or combination of networks
- Wireless communications
- Middleware services such as security and name servers
- Distributed database capabilities supporting both real-time and deferred data sharing
- Gateways to integrate legacy systems into your distributed environment

Servers use distributed database technology to share data in many different ways. These include remote data access and replication, which can be done on either a real-time or deferred basis.

This paper is about a fundamental component of a distributed computing solution: distributed database technology, which allows data to be shared between multiple servers. We will begin with an overview of key distributed database issues and Oracle's technology. We will then examine Oracle's latest distributed database technology—Oracle7 Symmetric Replication—in more depth.

Distributed Database Technology

Servers use distributed database technology to share data in many different ways. When a local server needs to share data that resides on a remote server, the local server can use distributed database technology to access the data on the remote system directly. Alternatively, the two servers can use distributed database technology to maintain copies, or replicates, of the data on both servers so that both can access it locally.

Added to this is the dimension of time. Do you need to access remote data in real-time or will deferred access suffice? Similarly, when replicated data is updated, do the updates need to be applied to other copies in real-time or can the updates be propagated on a deferred basis?

Real-time remote data access and real-time application of updates to replicated data is provided by synchronous distributed database technology. Deferred remote data access and deferred propagation of replicated data updates is provided by asynchronous distributed database technology. Understanding the tradeoffs between these two technologies is key to determining which to use to solve a particular business problem. Some business problems will require synchronous distributed technology. For others, asynchronous technology will be a better fit. In still other cases, both technologies will be needed to solve different aspects of the problem.

The tradeoffs between these two involve application integrity, complexity, performance, and availability. Synchronous technology ensures application integrity and minimizes complexity, but can be less available if the systems and networks involved are not reliable. It can also incur poor response time if network access between systems is slow. Asynchronous technology maximizes availability and minimizes response time, but can be more complex and requires careful planning and design to ensure application integrity.

Let's look at some examples.

Synchronous technology ensures application integrity and minimizes complexity, but can have poorer response time performance and less availability if the systems and networks involved are unreliable and slow.

Synchronous Distributed Technology Example

Say we have an order entry system and a shipping system. The systems share inventory data using synchronous distributed technology.

Since we are using synchronous technology, both systems have access to the most current, up-to-date inventory information. You could, for example, store a single copy of the inventory data on the shipping system and access the data remotely from the order entry system in real-time. Alternatively, you could use synchronous technology to maintain exact replicas of the inventory data on both systems. Any update to the data on either system would be applied synchronously, or in real-time, to the copy on the other system.

Orders can now be placed knowing exactly what is in inventory. Inventory items can be reserved as an integral part of the order-taking process. Shipments can be made knowing that the resulting decrements to inventory quantities will be immediately seen by the order entry system. In this way, synchronous technology provides simplicity and ensures application integrity.

To place orders or ship products, however, processing needs to take place across the network and on both systems, which may be slow. Also, the systems and network need to be reliable, or availability can suffer. If the shipping system or network goes down, for example, orders can't be placed. If remote access is being used, the order entry system can't access the inventory data on the shipping system. Similarly, if the inventory data is replicated, it can't be updated because the update can't be applied to both systems.

Asynchronous technology maximizes availability and response time performance, but can be more complex and requires careful planning and design to ensure application integrity.

Asynchronous Distributed Technology Example

Now let's use asynchronous distributed technology to share inventory data between our order entry and shipping systems. First, we will place the inventory data on the shipping system. Then we will use asynchronous replication to give the order entry system access to this information.

Updates to the inventory data on the shipping system will be propagated and applied to a copy of the inventory data on the order entry system, but on a deferred basis. This copy will contain accurate information but will not always be current. It may, however, be current enough for many businesses. If inventory levels are high and/or easily replenished, then slightly out-of-date information may suffice. The chances of ordering an item which is out of stock will either be very low or will only cause small shipment delays.

Building successful distributed computing systems is one of the great challenges of this decade. To ensure success, the distributed database technology that you select should be flexible, manageable, and integrated.

Ensuring that asynchronous distributed technology provides adequate application integrity requires careful consideration. The advantage of asynchronous technology, however, is availability and response time performance. Transactions operate against local data only. They may initiate deferred operations which need to be propagated to other systems, but if these other systems are not available the propagation will be deferred until the systems come back up. In our example, orders can be placed even when the network or shipping system is down.

Selecting Distributed Technology

Building successful distributed computing systems is one of the great challenges of this decade. To ensure success, the distributed database technology that you select should be:

Flexible—Distributed capability must fit your business—not the other way around. Whether an application requires remote access or replication, synchronous real-time operations or asynchronous deferred operations, the ability to query or update data or use procedure calls, the capabilities need to be there.

Manageable—Functionality without the ability to manage it is not usable. To minimize complexity, distributed operations should be as automatic and transparent as possible. Implementations should be robust to handle real-world stresses. Powerful management tools must be available.

Integrated—Distributed capability should be an integral component of the server. No additional, external components should be required that must be separately configured and administered. No performance penalty should be incurred requiring shared data to pass through extra components (extra “hops”) before it reaches the database where it is needed. No incompatibilities between the server and the external components should limit functionality.

Oracle7 Synchronous Capability

Oracle began offering synchronous distributed features in the mid-1980s. As Oracle's experience with distributed computing grew, Oracle's synchronous technology matured, culminating in the release of Oracle7. Oracle7 synchronous distributed technology is:

Oracle7 synchronous distributed technology is highly flexible. It supports distributed queries, distributed transactions, remote procedure calls, and synchronous replication.

Flexible

Oracle7 synchronous distributed technology meets the widest range of business needs.

Distributed Queries—Data on multiple databases can be queried using the full functionality of the SQL standard SELECT statement. This includes selection, join, aggregation, and sorting operations optimized for maximum performance in a distributed environment.

Distributed Transactions—Data on multiple databases can be modified using the full functionality of SQL standard UPDATE, DELETE, and INSERT statements operating as transactions to ensure that either all modifications on all databases complete successfully or are all rolled back should failures occur.

Remote Procedure Calls—Oracle7 servers can execute remote PL/SQL™ procedures on other servers. The remote procedure execution operates within the same transaction again ensuring that either all modifications complete or are all rolled back.

Synchronous Replication—PL/SQL triggers can apply all modifications to tables in one database directly to replicate copies of those tables in other databases. All modifications are applied within the same transaction to ensure exact, point-in-time consistency of all copies.

Oracle7 synchronous capabilities are manageable. Transparency minimizes complexity. A robust implementation protects against failures. Oracle Server Manager provides easy access to status information.

Manageable

Oracle7 synchronous distributed operations are automatic, robust and transparent to minimize complexity.

Location Transparency—Applications can access data and execute procedures remotely as easily as they do locally. No special coding is required to specify data or procedure location. Applications merely specify data and procedures using logical names. Mapping from logical names to physical locations is done transparently. Data and procedures can be moved from one database to another without modifying application code.

Distributed technology is built into the Oracle7 server. No additional, special servers need to be installed and maintained. Servers communicate efficiently using direct connections. Procedure calls work the same both locally and remotely.

Global Database Naming—Data can be uniquely identified and located in the distributed environment. External name services eliminate the burden of maintaining data location information within the database.

Commit Transparency—Applications can execute distributed transactions across multiple databases as easily as local transactions. The SQL standard COMMIT statement commits both local and distributed transactions transparently. No special coding is required.

Robust Protection Against Failures—Systems will fail and networks will fail. Oracle7 protects the integrity of distributed transactions automatically using a robust two-phase commit protocol. Complex two-phase commit logic does not need to be coded into applications.

XA Compliance—XA-compliant TP monitor facilities can be used to coordinate distributed transactions.

Status Monitoring—The status of distributed transactions can be easily obtained and monitored through standard data dictionary tables within the database using Oracle Server Manager™ and other tools.

Integrated

Oracle7 distributed technology is built into the Oracle7 server.

Single Server—No additional, special servers need to be installed and maintained.

Direct Server-to-Server Communications—Distributed operations are performed automatically through direct server to server connections. Applications do not need additional connections to external servers-to-perform operations such as distributed queries. Distributed data access does not need to pass through extra servers impacting performance.

Compatibility—Operations such as procedure calls operate under the same transactional protections whether they operate locally or remotely.

Symmetric Replication is Oracle's second-generation asynchronous distributed technology and the industry's most advanced.

Symmetric Replication is the most functional offering of asynchronous distributed capability available today.

Oracle7 Asynchronous Capability

Oracle introduced its first asynchronous distributed capability in the initial Oracle7 release in late 1992. This feature, called read-only snapshots, provides a basic asynchronous replication capability. One site, the snapshot master, can be updated. All other replicates or snapshots are read-only. Incremental row changes are propagated on demand or at time-based intervals using a fast refresh mechanism. The snapshot refresh group feature ensures transactional consistency to maintain referential integrity between multiple snapshots. Read-only snapshots are also very easy to create and administer.

Based on the experience gained from this first offering, Oracle now introduces its next-generation asynchronous distributed technology—Symmetric Replication—the industry's most advanced. Oracle7 Symmetric Replication is:

Flexible

Oracle7 Symmetric Replication provides the most functional offering of asynchronous distributed capability available.

Basic and Advanced Replication Support—Applications can implement primary-site, dynamic, and shared-ownership models as well as fail-over configurations (see below).

Full Transactional Consistency—The referential integrity of replicated data is ensured.

Automatic Update Conflict Detection and Resolution—Resolution routines can be selected declaratively from a set of predefined standard routines such as most recent timestamp or site priority. Users can also define their own customized resolution rules.

Full and Subset Table Replication—All rows or only selected rows in a table can be replicated.

Event-and Demand-Based Replication Methods—Replicated row changes can be efficiently propagated either immediately or when they are demanded by the target system.

Deferred Remote Procedure Calls—Remote PL/SQL procedures can be executed in an asynchronous, or store-and-forward, manner.

Oracle provides powerful management tools integrated into the database and as outside GUI-based client applications.

Manageable

Oracle7 Symmetric Replication provides powerful management tools integrated into the database and as outside GUI-based client applications.

Replication Catalog—Provides a single, consolidated repository of meta data that defines the distributed/replicated environment, i.e., what database objects (tables, procedures, triggers, indexes, etc.) are replicated where and how they are being replicated. The replication catalog is itself replicated to multiple sites to ensure high availability and easy local access to authorized users.

Distributed Schema Management—Allows replicated environments to be defined and changed automatically at multiple sites by replicating and applying data definition language (DDL) commands. For example, operations such as adding an index or check constraint to a table everywhere it is replicated can be done automatically without tedious and error-prone manual operations.

SNMP Replication MIB—Allows system monitoring tools supporting the SNMP industry standard to monitor Oracle7 Symmetric Replication; Defined as a Management Information Base (MIB) which extends the standard RDBMS MIB jointly defined by Oracle and other database vendors.

Oracle Server Manager—New Oracle Server Manager extensions provide an easy to use, GUI-based administration capability for Oracle7 Symmetric Replication. Administrators can easily query the replication catalog, examine internal replication engine components, initiate distributed schema management operations, and troubleshoot problems. Server Manager can be launched in context from system monitors and other tools.

SMTI Member Administration Tools—Additional distributed systems management tools supporting Oracle7 Symmetric Replication are available from Oracle partner companies supported through Oracle's SMTI (System Management Tools Initiative) program. Oracle7 Symmetric Replication users can continue to use the monitors and other administration tools they already have deployed.

Oracle7 Symmetric Replication is built into the Oracle7 server as an internal, integrated facility to further improve manageability, maximize performance, and ensure compatibility with other database features.

With external replication server-based products, contention for log access can bottleneck systems even when only a relatively small amount of data is actually being replicated.

Integrated

Oracle7 Symmetric Replication is built into the Oracle7 server as an internal, integrated facility to further improve manageability, maximize performance, and ensure compatibility with other database features.

Integration further improves manageability.

Single Server—No additional, external components must be configured, monitored and maintained.

Standard Components—Oracle7 Symmetric Replication is implemented using proven database components such as tables, views, and PL/SQL procedures that Oracle users are already familiar with.

Standard Backup/Recovery—Oracle7 Symmetric Replication is protected by Oracle7's standard backup and recovery mechanisms. No additional complex procedures involving extra external components are required. Database systems can be backed-up and recovered separately without the need to synchronize operations with other sites.

Integration maximizes performance:

Efficient Implementation—Oracle7 Symmetric Replication uses built-in, high-performance database components such as stored procedures and triggers to capture, propagate, and apply replicated row changes.

Direct Server-to-Server Propagation—Replicated data is propagated directly from server to server. There are no intermediary components to pass through, i.e., no extra “hops.”

Proportional Processing—Replication processing is proportional to the amount of data that is actually being replicated, not to total system throughput. No “replication cost” is paid when updates are performed against tables that are not being replicated.

No Log Contention—No contention for log access can bottleneck high-throughput systems.

External replication server-based products must process each and every row change written into the log file to determine whether the change needs to be replicated or not. Contention for log access can bottleneck systems even when only a relatively small amount of data is actually being replicated. This problem is exacerbated because the server needs to be modified to write out a substantial amount of extra information for replication that is not needed for recovery purposes.

Integration ensures compatibility with other database features.

National Language Support—Oracle7 Symmetric Replication uses and is fully compatible with the Oracle7 National Language Support facility.

Parallel Data Query—Oracle7 Symmetric Replication is fully supported on systems using Oracle7 parallel data query capabilities.

External replication products only provide national language support and other standard database server capabilities when they reimplement the capability into the external components. Because of this, replication functionality tends to lag server functionality. Incompatibilities may also arise when new capabilities are provided by still other external components. For example, parallel features, such as parallel query capabilities, provided as an external mechanism outside the server may be incompatible with the external mechanisms supporting replication.

Replication Concepts

Replication is a key element of an overall distributed computing solution, but also a demanding one. One particular challenge is understanding the range of potential uses for synchronous and asynchronous replication.

With synchronous replication, all copies of data are kept exactly synchronized and consistent. If any copy is updated, the update is immediately applied to all other copies within the same transaction. Synchronous replication is appropriate when this exact consistency is important to the business application.

With asynchronous replication, copies or replicates of data will become temporarily out of sync with each other. If one copy is updated, the change will be propagated and applied to the other copies as a second step, within separate transactions that may occur seconds, minutes, hours, or even days later. While copies can be temporarily out of sync, over time the data should converge to the same values at all sites.

Ensuring convergence in asynchronous replication environments is critical for nearly every application. What happens, though, if the same data element, e.g., the same column in the same row, is updated at two sites at the same time, or to be more precise within the same replication interval? This is known as an update conflict. To ensure convergence, update conflicts must be detected and resolved, so that the data element has the same value at every site. Alternatively, update conflicts may be avoided by limiting “ownership,” or the right to update a given data element, to a single site.

To ensure convergence in asynchronous replication environments, update conflicts must be detected and resolved or they must be avoided. Oracle7 Symmetric Replication supports both of these approaches.

Usage Models

Asynchronous replication usage models are methodologies for conflict avoidance or conflict detection and resolution. Applications employing asynchronous replication use one or more of these methodologies to ensure data convergence. Applications include the logic necessary to understand, for example, that a conflict avoidance methodology is being employed and what data is owned and updatable at a given site. Similarly, if the business requirement demands that ownership be shared across multiple sites, the application needs to take into account the potential for update conflicts and their detection and resolution.

Oracle7 Symmetric Replication is designed to support replication models employing both conflict avoidance and conflict detection and resolution. Unlike other replication products, the underlying replication mechanisms do not limit users to a single replication model. It is called *symmetric* because all replicates are potentially updatable. Applications employing conflict avoidance models restrict updates to certain sites to prevent conflicts. Optionally, database facilities such as views and triggers can be defined as “guards” to enforce exclusive ownership. For shared-ownership models, Oracle7 Symmetric Replication provides built-in, automatic conflict detection and resolution.

In the following section, we will describe in more detail two methods of conflict avoidance: primary-site ownership and dynamic ownership. We will then discuss the shared-ownership model and issues involving conflict detection and resolution. Finally, we will discuss the use of replication for fail-over configurations which combine elements of both conflict avoidance and conflict detection/resolution. We will provide examples of how each of these models can be applied to solve real business problems.

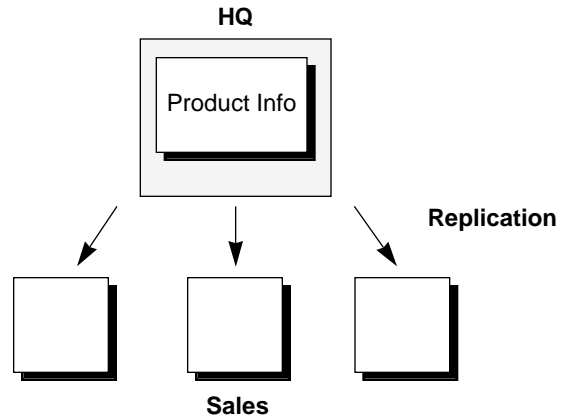
Primary-Site Ownership

With primary-site ownership, asynchronously replicated data is “owned” by one site. Ownership means that the site may update the data. Other sites “subscribe” to the data owned by the primary site, which means that they have access to read-only copies on their local systems.

Primary-site replication has many uses, for example:

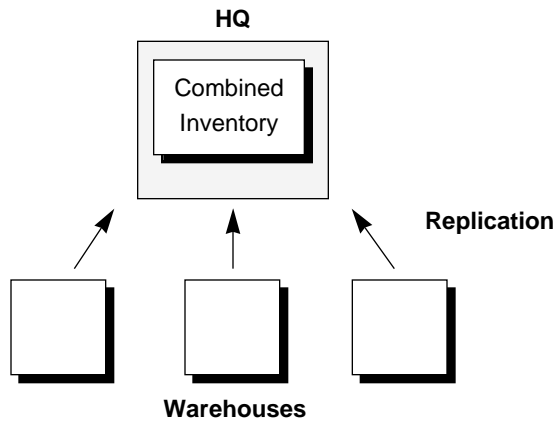
Distribution of Centralized Information—Product information such as price lists can be maintained at a corporate headquarters site and replicated to read-only copies maintained on order entry systems at remote sales offices.

Information Distribution



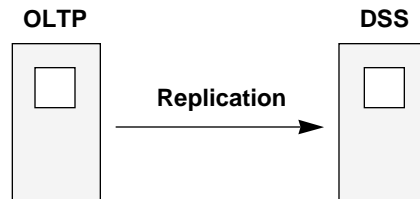
Consolidation of Remote Information—Inventory data maintained on systems in a number of remote warehouse locations can be replicated to a consolidated read-only copy of the data at a corporate headquarters site.

Information Consolidation



Offloading of OLTP Data for Decision Support System (DSS) Analysis—Data from one or more OLTP systems can be offloaded to a separate, local DSS for read-only analysis.

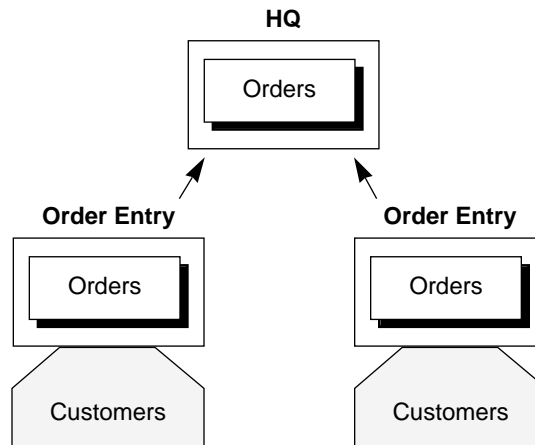
Information Offloading



A primary site may own the data in an entire table, in which case other sites subscribe to read-only copies of all or some subset of that table. Alternatively, multiple sites may own distinct subsets or partitions of the table. Each site might own a distinct set of rows, i.e., a horizontal partition, or a distinct set of columns, i.e., a vertical partition, within a table. Other sites then subscribe to read-only copies of all or some further subsets of the partitions.

For example, a distributed order entry system could be implemented such that each order entry site in each sales office owns distinct horizontal partitions of tables, e.g., CUSTOMERS, ORDERS, and ITEMS tables, that contain the customer and order information for the customers serviced by each office. A central headquarters site could subscribe to the data owned by each order entry site to maintain a consolidated read-only copy of all orders and customer information across all the sales offices.

Partitioned Primary Site Order Entry

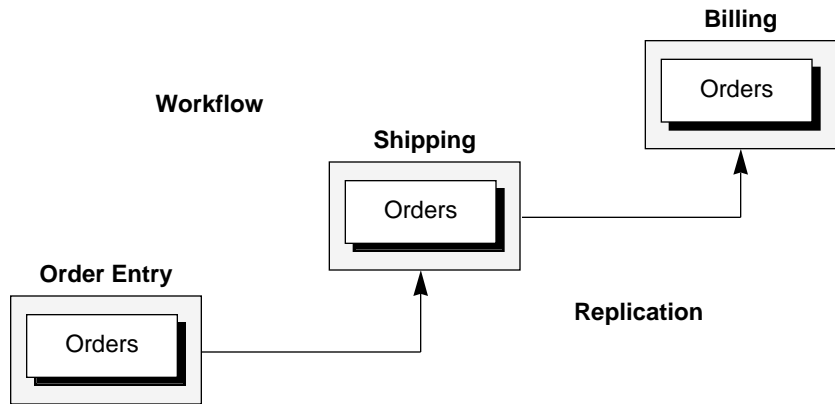


Dynamic Ownership

With dynamic ownership, the ability to update asynchronously replicated data moves from site to site while ensuring that only one site may update the data at any given point in time. For example, within an order processing system, the processing of orders typically follows a well ordered series of steps, e.g., orders are entered, approved, shipped, billed, collected, accounted for, and so on. Centralized systems allow the application modules that perform these steps to act on the same data contained in one integrated database. Each application module acts on an order, i.e., performs updates to the order data, when the state of the order indicates that the previous processing steps have been completed. For example, the application module that ships an order will do so only after the order has been entered and approved.

By employing a dynamic ownership replication methodology, such a system can be distributed across multiple sites and databases. Application modules can reside on different systems. For example, order entry and approval can be performed on one system, shipping on another, billing on another, and so on. Order data is replicated to a site when its state indicates that it is ready for the processing step performed by that site. Data may also be replicated to sites that need read-only access to the data. For example, order entry sites may wish to monitor the progression of processing steps for the orders they enter.

Dynamic Ownership Order Processing System



Shared Ownership

The replication models described thus far, primary site and dynamic ownership, share a common property: at any given point in time, only one site may update the data while the other sites have read-only access to replicated copies of the data.

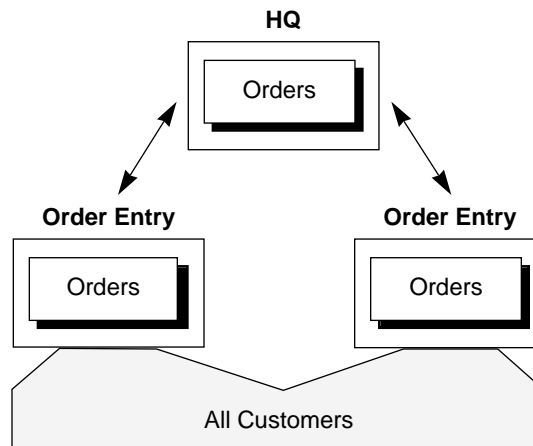
In some situations, however, it is desirable to allow multiple sites to update the same data, potentially at the same time. For example, it may be desirable to replicate customer data across multiple sites and systems rather than maintain customer data centrally. In this case, different sites may need to update this data.

Update Conflicts

Say we are replicating customer data across sales office order entry sites and headquarters sites. One element of the customer data is the customer address. What happens if a customer's address is changed at both a sales office and a headquarters site at the same time?

This occurrence is known as an *update conflict*—asynchronously replicated data has become inconsistent because the replicated data can be updated at multiple sites. For some applications, though, temporary inconsistencies can be permitted as long as they can be detected and resolved to ensure that the replicated data will converge to a consistent state at all sites.

Shared Ownership



Sophisticated Uses of Shared Ownership

Shared ownership allows asynchronous replication to be employed where primary-site and dynamic-ownership models would be too restrictive. In cases where temporary inconsistencies can be permitted and conflict detection and resolution employed, it can offer unprecedented new capability.

For example, earlier we discussed how a distributed order entry system could be implemented using primary-site replication methodologies with horizontal partitioning. In this scenario, each sales office owned a distinct horizontal partition of the tables containing orders and customer information for the customers serviced by each office. Each sales office entered orders for its customers, but no others.

For some businesses, though, this is not the model. For example, a retail chain may have several stores in a metropolitan area. Customers may frequent the store closest to where they live, but go into other stores which will also want to take their orders. If multiple stores perform updates to the same customer and order data, however, update conflicts can occur. Sophisticated application developers can identify these conflicts and either select standard resolution routines or devise their own.

Fail-Over

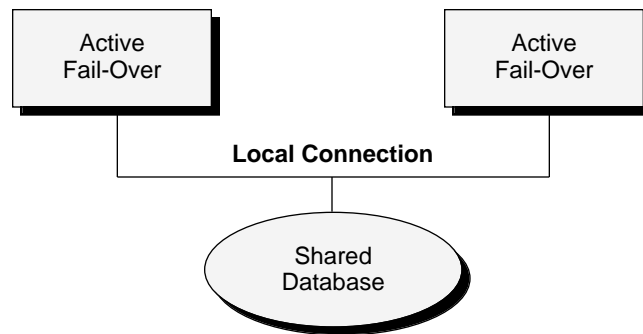
Oracle7 Symmetric Replication is one of many technologies and approaches that can be employed to support fail-over configurations. A fail-over configuration protects against failures of a primary system by enabling continued processing on a secondary, fail-over system. Asynchronous replication can be used to maintain copies of a primary system's data on a fail-over system. Processing can continue against this replicated data if the primary system fails.

Oracle also provides two other options that should be considered when selecting a fail-over solution. These are: 1) the Oracle Parallel Server,[™] and 2) a standby database configuration. These options offer you alternative tradeoffs in terms of throughput capacity, ease of operation, potential for transaction loss and other data inconsistencies, limitations on the uses of the fail-over system, and the type of failure conditions that can be protected against.

Parallel Server

The Oracle Parallel Server provides fail-over capabilities in locally connected cluster or massively parallel environments. In these environments, it will usually be the preferred option.

Oracle Parallel Server



The Oracle Parallel Server allows multiple systems to share common access to a single database maintained on shared disk devices. If a system fails, an Oracle Parallel Server instance on one of the surviving systems will automatically recover any incomplete transactions. Applications that were running on the failed system can then resume processing on a surviving system against the same database.

The Oracle Parallel Server offers high-throughput capacity, ease of operation, and no potential for transaction loss. All systems in an Oracle Parallel Server environment can be fully utilized for both query and update access at any time. Oracle Parallel Server, however, does not provide protection against site failures that may render the entire locally connected environment inoperative, e.g., failures that might be caused by a power outage, fire, flood, or sabotage.

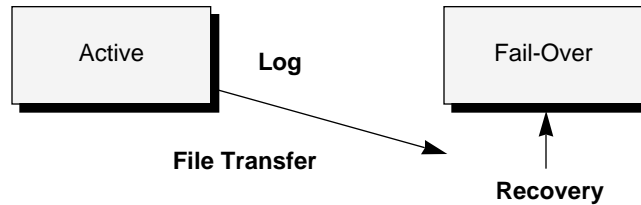
Standby Database Configuration

Another fail-over option is a standby database configuration. This approach protects against site failures and also offers high-throughput capacity. It uses an alternative form of asynchronous replication which

employs operating system file transfer facilities to move the log files generated by the primary system to a fail-over site. The log files are then applied to the fail-over system using Oracle recovery mechanisms.

The fail-over system may be located in a geographically remote area to protect against site failures at the primary system's location. High throughput volumes are supported because file transfer does not incur log contention with the primary server and Oracle's recovery mechanisms are highly efficient.

Standby Database Configuration



The fail-over system is maintained in a continuous recovery mode which means that it cannot be used for purposes other than fail-over. Also, if the primary system's site should fail, it may not be possible to transfer the most recently generated log file or files to the fail-over site. Any transactions contained in these log files may be lost.

Oracle currently offers this solution as an Oracle Consulting service. It will be offered as a supported product feature later in 1995.

Oracle7 Symmetric Replication and Fail-Over

Oracle7 Symmetric Replication can also be used in a fail-over configuration to protect against site failures. Its advantage over the standby database approach is that the fail-over system or systems can be used for purposes other than fail-over.

Oracle7 Symmetric Replication



Data that is being updated on the primary system can be queried on the fail-over system at any time, e.g., for reporting or decision support analysis. The same data on both the primary and fail-over systems can also be updated at any time if the two systems are employing a shared-ownership replication model with full update conflict detection and resolution.

The throughput capacity is less than the standby database approach because Oracle7 Symmetric Replication does not use the Oracle recovery mechanisms to apply replicated data changes. Symmetric replication, like all fail-over solutions based on any form of asynchronous replication, also has the potential for transaction loss in some failure scenarios. Because of its asynchronous or store-and-forward nature, it is possible for replicated changes to have been stored on the primary system but not yet forwarded to the fail-over site when the failure occurs. These stored transactions may not be recoverable.

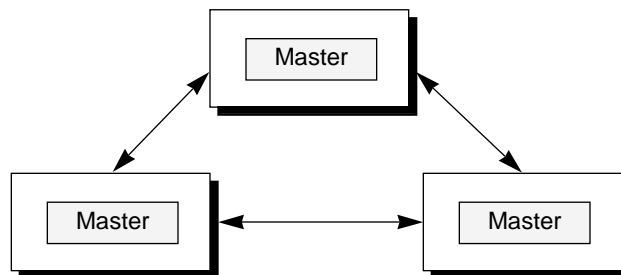
Configurations

Oracle7 Symmetric Replication supports replication of both full tables and subsets of tables through two mechanisms: multiple masters and updatable snapshots. These two mechanisms can be combined in hybrid configurations to meet different needs.

Multiple Masters

Multiple master replication supports full table, peer-to-peer replication between master tables. Master tables at all sites can be updated. Changes applied to any master table are propagated and applied directly to all other master tables. Failures of any one master site will not block propagation of changes between other master sites.

Multiple Master Replication

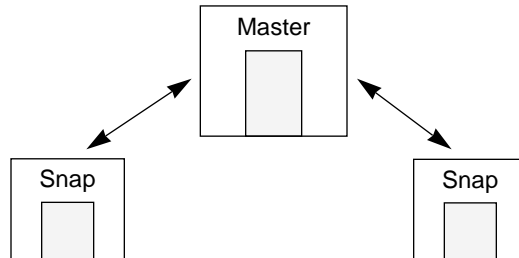


Multiple master replication uses deferred RPCs (described below) as the underlying transport mechanism to propagate and apply changes. Changes to multiple master tables are applied in a transactionally consistent manner to ensure data and referential consistency. Changes are propagated either immediately in an event-based manner, or at specified points in time when connectivity is available or when communications costs are lowest, e.g., during evening hours. If a remote system is unavailable, the deferred RPCs propagating changes to that system remain in their local queue for later execution.

Updatable Snapshots

Oracle has extended the initial Oracle7 snapshot mechanism to support Symmetric Replication. Snapshots, as well as the snapshot masters, can now be updated. Updates to snapshots are propagated and applied to snapshot masters using deferred RPCs as the underlying mechanism.

Updatable Snapshots



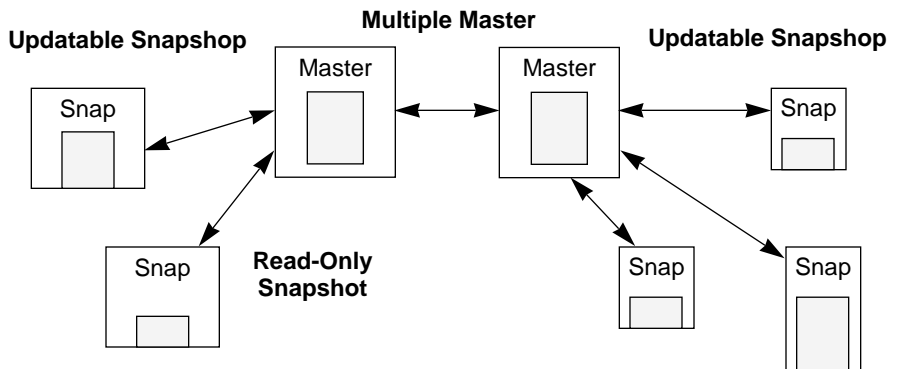
Snapshots can be defined to contain a full copy of a master table or a defined subset of the rows in the master table that satisfy a value-based selection criterion. Snapshots are refreshed from the master at time-based intervals or on demand. Any changes to the master table since the last refresh are then propagated and applied to the snapshot. Multiple snapshots are refreshed in a transactionally consistent manner to ensure data and referential integrity.

Hybrid Configurations

Multiple master replication and updatable snapshots can be combined in hybrid configurations to meet different needs. Specifically, snapshot masters can be replicated in multiple master configurations. This allows full-table and table subset replication to be combined in one system.

For example, multiple masters can replicate between two hub sites supporting two geographic regions. Read-only or updatable snapshots can be defined on the masters to replicate full tables or table subsets to sites within each region. This configuration allows the two master sites to function as fail-over sites for each other. An added benefit of this configuration is that snapshots can be re-mastered from the other hub site to provide an added measure of high availability.

Hybrid Configuration



Oracle7 Symmetric Replication automatically detects update conflicts and invokes user-specified conflict resolution routines to restore consistency and ensure data convergence.

Conflict Detection and Resolution

Oracle7 Symmetric Replication automatically detects update conflicts and invokes user specified conflict resolution routines to restore consistency and ensure data convergence. Conflicts are automatically detected by comparing the before image of the originally modified data to the current values of the data at replicate sites. If the values are different, a conflict has occurred.

Oracle7 Symmetric Replication includes a set of predefined, standard conflict resolution routines that users can declaratively select. Optionally, users can define their own customized routines. Predefined, standard routines include:

- Latest timestamp
- Earliest timestamp
- Priority group
- Site priority
- Additive
- Minimum
- Maximum
- Average
- Overwrite
- Discard

Users can define their own customized resolution routines using PL/SQL procedures. These customized routines can implement more application specific business rules. They might also invoke notification routines, e.g., generate e-mail messages or inserts into a transaction history table to notify users that a conflict occurred and how it was resolved.

Conflict resolution can be configured to invoke multiple resolution routines based on user-specified priorities. For example, a resolution routine could be specified to always execute first. If that routine is not able to resolve the conflict due to special circumstances that it detects, another routine can be specified to run second to handle those special circumstances.

Conflict detection and resolution is needed for the shared ownership replication model discussed above. It is also needed for dynamic ownership and fail-over models in certain configurations.

Column Groups

Conflict detection and resolution is based on column groups. A column group is a collection of logically related columns within a row that need to be dealt with as a unit to ensure consistency. An example would be a

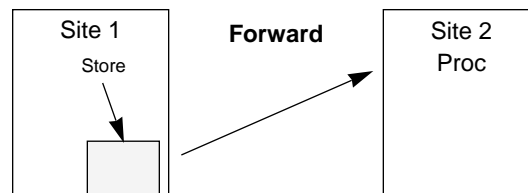
set of columns used to contain addresses in a customer table. If address information needs to be updated at multiple sites, a latest timestamp resolution method could be selected to ensure that the most recent address change is maintained should conflicts occur.

A row can have multiple column groups. For example, a column in the same customer table might contain information about customers' outstanding credit balances. This column may be incremented by a billing system and decremented by an accounts receivable system at different sites. Here, the additive resolution method could be used to resolve conflicts.

Deferred RPCs

Deferred RPCs are a flexible, general purpose facility. They are used as a propagation mechanism for replication. The facility is also available for direct use to enable calls to remote PL/SQL procedures to be processed in an asynchronous or store-and-forward manner.

Deferred RPCs



A local transaction initiates the execution of a deferred RPC by submitting a request to a local queue. Submission into the queue is done within the local transaction. Entries in the queue are then pushed to their target location(s) and executed as a second step within separate transactions.

If a remote system is unavailable when the deferred RPC queue is pushed, the entries for that target system remain in the queue for later propagation. The deferred RPC queue is durable, protected by the backup and recovery mechanisms of the Oracle7 server. This guarantees that the request will not be lost and can be propagated and executed when the target system becomes available.

Unlike other Oracle's vendors' offerings, deferred RPCs can be easily targeted to one or multiple remote systems. Also, multiple deferred RPCs submitted with the same local transaction are remotely executed together within the same transaction without requiring special coding.

Using Deferred RPCs

Deferred RPCs have many uses. Earlier we discussed how asynchronous replication could be used to propagate orders from an order entry system to a remote shipping system. Replication maintains copies of the orders on both systems. For some businesses, though, it may not be necessary to maintain copies of orders on the order entry system. All that is needed is a reliable, asynchronous mechanism to forward the orders to the shipping system.

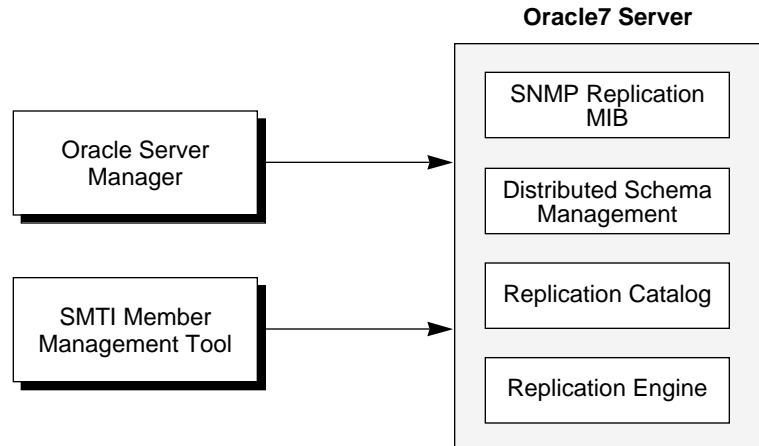
For example, a department store chain may perform all order processing, inventory, billing, distribution and other functions from a central processing location. At each store, a mechanism is needed to forward information about purchases to this central facility. There is no need, however, to maintain copies of this purchase information locally in each store. In such a situation, purchases could be forwarded from the store to the central facility using deferred RPCs. The deferred RPC mechanism guarantees delivery of the purchase information which is all the functionality that is required.

Procedural Replication

Deferred RPCs are also used for procedural replication. Occasionally it may be necessary to perform large numbers of updates in a serial, batch-oriented manner to replicated data. For example, perhaps once a quarter or once a year it may be necessary to purge old order data that may no longer need to be kept online. Replicating each individual row change across multiple sites could be very inefficient. Instead, a procedure can be executed at each replicate site to perform the updates directly. The distributed schema management facility (discussed in the next section) provided with Oracle7 Symmetric Replication allows you to easily set up and maintain remote procedures at multiple sites for procedural replication.

Management Tools

Management tools are critical to successful distributed computing. Advanced replication functionality without the ability to manage it is not usable. Oracle provides management tools integrated into the database and as outside GUI-based client applications as is appropriate for the functionality they provide. Open interfaces and Oracle's SMTI program enable other vendors to enhance their management tool products to support Oracle7 Symmetric Replication.



Replication Catalog

The replication catalog provides a single, consolidated repository containing the meta data that defines the replicated environment. The replication catalog is itself replicated to multiple sites to ensure high availability and easy local access by authorized users wherever they may be located.

The replication catalog defines the database objects being replicated, the sites where they are replicated, and the mechanisms used to support their replication. Database objects include the tables that contain replicated data and the object definitions for other supporting components including indexes, views, procedures, triggers, and synonyms. An extension to Oracle's standard data dictionary, this meta data is open and accessible. It can easily be queried using standard SQL.

Other vendors who provide replication using an external replication server-based implementation do not provide a single, consolidated repository. Meta data and control information is scattered across multiple separate external servers, providing no system-wide view. Each replication server only maintains the information necessary for its operation.

Distributed schema management allow replicated environments to be defined and changed from a single control point by automatically replicating and applying data definition language (DDL) commands to multiple sites.

Distributed Schema Management

Oracle7 Symmetric Replication's distributed schema management capability allows replicated environments to be defined and changed from a single control point. It automatically replicates and applies data definition language (DDL) commands to multiple sites. It also automatically generates the underlying mechanisms using database triggers and procedures to support replication.

Distributed schema management allows you to perform operations such as:

- Add an index or check constraint to a replicated table everywhere
- Add all the database objects in a replicated schema to a new site
- Alter a procedure or view definition everywhere

Distributed schema management operations are performed and controlled from one site called the master definition site. It automatically pushes DDL to all master sites and allows snapshot sites to pull down and apply DDL on demand. Because it uses and maintains the replication catalog, which is itself replicated to multiple sites, the master definition site can be changed in case of failure. Distributed schema management operations are initiated by GUI-based management tools from Oracle or other vendors through an open procedural API.

With other replication products, distributed schema management is a manual operation requiring system administrators to connect to multiple databases, replication servers, and other external components to execute DDL and configuration procedures. Configuration and maintenance of distributed schemas can involve hundreds of such operations. When done manually, this is complex, tedious, and highly error prone.

Symmetric Replication can be monitored using the industry-standard SNMP protocol.

SNMP Replication MIB

Oracle7 Symmetric Replication can be monitored using the industry-standard Simple Network Monitor Protocol (SNMP). The Symmetric Replication MIB, or Management Information Base, extends the standard RDBMS MIB jointly defined by Oracle and other database vendors. SNMP-based tools such as HP's OpenView, Sun's SunNet Manager, and IBM's NetView/6000 can be used to monitor replication, database, operating system, and network activity from a single console. You can continue to use the monitoring tools you already have in place. Other replication products require that you install and maintain an additional monitoring system just for replication.

Oracle Server Manager provides an easy-to-use, GUI-based administration capability for Oracle7 Symmetric Replication.

Oracle Server Manager

Oracle Server Manager provides an easy-to-use, GUI-based administration capability for Oracle7 Symmetric Replication as an extension to its capabilities for general database administration, and can be launched in context by other administration tools including system monitors.

Oracle Server Manager allows system administrators to easily:

- Query the replication catalogs and display SNMP replication MIB status information
- Examine internal replication engine components such as the deferred RPC queue and error log
- Initiate distributed schema management operations and monitor their execution across multiple sites
- Troubleshoot problems

SMTI Member Administration Tools

Additional distributed systems management tools are available from or under development by third-party management tool vendors. Oracle supports and encourages these third-party providers through its SMTI program. Oracle's SMTI program includes over 200 products from over 75 vendors, providing a wide range of administrative capabilities for Oracle.

The SMTI program fosters a close working relationship between Oracle and providers of complementary administration tools by providing:

- Early access to new product releases
- Technical assistance and training
- Cooperative development partnerships
- Joint support arrangements providing customers with a single point of contact

Oracle7 Symmetric Replication was designed to support open interfaces to further encourage SMTI tool development. These include:

- The SNMP replication MIB
- The Oracle7 Symmetric Replication procedural API for distributed schema management
- Standard SQL query access to the replication catalogs
- Standard SQL query access to underlying replication engine components such as the deferred RPC queue and error log implemented using standard relational tables

Packaging

Oracle7 Replication is available in two packages. The first is the read-only snapshot feature which provides basic replication capabilities and is available with the Oracle7 Distributed Option. The second is the full capability of Symmetric Replication, as described herein, which is available with the Oracle7 Advanced Replication Option. The Advanced Replication Option requires the Distributed Option as a prerequisite. Oracle also recommends the Oracle Education and Consulting services available for Symmetric Replication.

Conclusion

Building integrated, effective distributed computing systems is one of the great challenges of this decade. Distributed database technology is a fundamental component of a complete distributed solution. Oracle7 distributed database technology is:

Flexible to let you build distributed solutions that meet your business requirements. Whether your application requires remote access or replication, synchronous real-time operations or asynchronous deferred operations, the ability to query or update data or use procedure calls, the capabilities are there.

Manageable so that distributed systems are practical. Management tools, robust implementations to handle real-world stresses, and automatic and transparent features keep the complexity of distributed operations under control.

Integrated to improve manageability, maximize performance, and ensure compatibility with other database features. Oracle7 distributed technology is built-in—not added-on.

Oracle7 offers the most advanced distributed database technology in the industry. From the mid-1980s when Oracle introduced its first distributed capabilities, to today with the introduction of Symmetric Replication, Oracle has set the pace. Oracle's experience, resources and technical leadership ensure your success both now and in the future.

Author: Gordon Smith

Contributors: Benny Souder, Alan Downing, Dean Daniels,
Gary Hallmark, Ken Jacobs, Sandeep Jain, Merrill Holt, Scotty Nath

This document is provided for informational purposes only and the information herein is subject to change without notice. Please report any errors herein to Oracle Corporation. Oracle Corporation does not provide any warranties covering and specifically disclaims any liability in connection with this document.

Oracle is a registered trademark, and Oracle7 are trademarks of Oracle Corporation.


ORACLE®

**Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
USA**

**Worldwide Inquiries:
415.506.7000
Fax 415.506.7200**

**Copyright © Oracle Corporation 1995
All Rights Reserved
Printed in the U.S.A.**

Part #: A33128

 **Printed on recycled paper.**