Altibase White Paper

Replication Services

ALTIBASE® HDB™ Hybrid RDBMS

Publication Date: May 2011
Author: Cy Erbay, Principal Technologist, Altibase, Inc.

Table of Contents

Introduction

As today's corporations process more and more data and work towards integrated IT systems across multiple business units, the impact of system outages becomes more severe and less tolerable. Corporations face revenue loss and security threats. Customer confidence erodes when information is unavailable or inaccurate. Data accessibility and quality issues can prevent businesses from meeting their objectives.

Traditionally, IT organizations have focused their high availability planning and related technology deployments on preparations for disasters or other unplanned outages. However, operations may be impacted significantly by frequent interruptions, such as planned outages or performance problems.

Consequently, IT departments of corporations need to adopt technology solutions that eliminate all causes of downtime and data loss while improving system performance. They demand database software vendors to provide replication and load-balancing services for their mission critical applications.

ALTIBASE HDB meets high-availability and fault tolerance requirements of today's modern IT environments with its built-in replication feature which supports implementations with redundancy over TCP/IP networks, supporting up to 32 hot swappable replicated nodes, granular and selective replication of databases at table unit levels at very high speeds.

This white paper is provided for those who desire a deeper understanding of how ALTIBASE HDB Replication feature is designed and how it functions.

ALTIBASE HDB Replication Key Features

ALTIBASE HDB combines several key services such as high-availability, fault tolerance and load balancing with its built-in replication feature.

ALTIBASE HDB replication feature maintains an up-to-date backup of the database on an active server, and in the event of that server unexpectedly becomes unavailable, immediately resumes services again from an identical database on an alternate server, providing a non-stop operating environment.

ALTIBASE HDB replication is a TCP/IP based service. It supports up to 32-way replication (hot swappable replicated nodes). The replication environment can include remote servers geographically distributed across different locations. In addition, replication is supported in a multiple IP network environment. In other words, it is possible to perform replication between two hosts having two or more physical network connections using multiple IP addresses.

Another benefit of the ALTIBASE HDB replication feature comes from its ability to perform database replication at the table unit level. Users can perform replication against select tables without worrying about replicating an entire database. Users also have the ability perform add/drop table type of tasks during replication without stopping the service.

ALTIBASE HDB's log-based replication architecture, while providing very speedy performance, imposes very little overhead on computing resources since it only transforms transaction logs into logical logs and sends them to remote servers for processing.

ALTIBASE HDB replication provides users with a choice of two commonly used modes of replication: Lazy mode (asynchronous) and Eager mode (synchronous). Lazy mode focuses on high performance while Eager mode focuses on data integrity and consistency.

In replication environments, it is quite common to have data conflict issues. ALTIBASE HDB provides a built-in audit feature that can discover and auto-resolve data conflict issues due to replication.

To tackle issues of data integrity, ALTIBASE HDB replication also provides an offline replication option. With this option, a receiver node can directly access transaction logs of a sender node.

ALTIBASE HDB also makes use of separate threads for detecting network failures.

ALTIBASE HDB replication allows replication among heterogeneous HDB platforms that may be running on different CPU and operating systems.

The computing environment can maintain near standalone performance during replication operations (90% Active-Active and 96% Active-Standby).

ALTIBASE HDB Replication

The primary purpose of database replication is to maintain an up-to-date backup of data on an active server and provide an uninterrupted service environment in which an alternative server can be used to resume service in the event the active server unexpectedly goes offline or becomes unavailable.

The basic idea behind replication in ALTIBASE HDB is the use of a log replay method. With this method, a sender node transforms database transaction logs to change logs, internally known as XLOGs, and sends them to a receiver node. The receiver node creates the same exact database transactions using XLOGs it has received. In other words, only XLOGs are additionally maintained in order to propagate changes to a remote server,

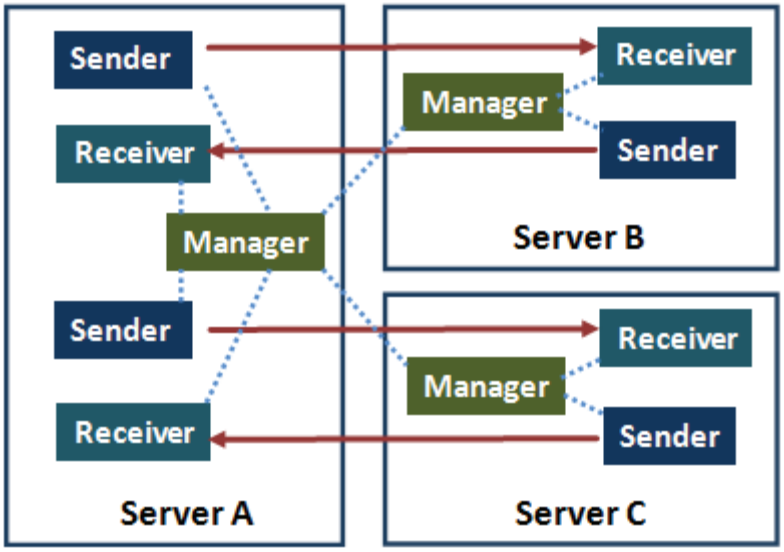

Figure 1: Altibase HDB Replication Thread Architecture

ALTIBASE HDB replication feature is implemented with thread architecture. There are four major thread components; sender, receiver, manager and heartbeat. The sender and receiver threads have the responsibility of transporting XLOGs, while the manager has the role of arbitration between the two. The heartbeat thread allows each node to check the condition of other nodes in the replication environment to detect failure conditions at user configurable time intervals.

When a replication service is started, a local replication manager creates a sender thread, and this thread connects to a replication manager on a remote server. The replication manager on the remote server then generates a receiver thread.

A sender thread on a local server sends information about changed database contents to a remote server, and where a receiver thread applies the same changes to its database. Additionally, sender and receiver threads automatically detect whether corresponding servers shut down normally or abnormally, and then perform appropriate tasks.

When transactions occur, ALTIBASE HDB records the information necessary for transaction processing and data recovery. This information is recorded in transaction logs. Sender threads use the transaction logs to find out what type of transactions occurred, or what the data values were before and after the modifications took place.

Sender thread creates XLOGs from the transaction logs for transmitting to receiver threads of remote nodes in the replication environment.

An XSN (XLOG Sequence Number) is used to track the final position for a change log that has been transmitted by a sender thread to a receiver thread. XSNs are maintained on replication metadata tables by both sender and receiver threads on each side of the replication.
An XLOG contains the following information:

Table – used to identify a target table
Primary Key – used to identify a target record
Column – used to identify a target field
Before Value - used for comparing data between sender and receiver
After Value – used to apply data on a receiver node

ALTIBASE HDB replication ensures high transaction processing performance in a sender node since there is no interference (or lock) between HDB service threads and the replication sender

threads in the node. In other words, the normal database transactions and the replicated transactions take place in complete isolation from one another.

A receiver thread receives XLOGs from a sender thread and calls the receiver-applier module for processing. Because XLOGs are already validated in the sender node, the applier module on the remote node can request their processing by HDB Storage Manager Module without further validation.

After the receiver thread on the receiver node is done applying XLOGs, it sends the acknowledgment information (ACK) to the sender thread, indicating that XLOGs are already applied in the receiver node.

In the event of a communication failure between sender and receiver nodes, the sender node continuously checks the communication status until the network is recovered. Once the network is recovered, the sender thread connects to the receiver thread of the receiver node, and re-transfers XLOGs to the receiver node.

ALTIBASE HDB Replication Modes

In today's business environment, IT departments need to ensure high performance and high availability at the same time.

ALTIBASE HDB replication provides users with choice of two modes of replication: Lazy (asynchronous), and Eager (synchronous). Lazy mode focuses on high performance while Eager Mode focuses on data integrity and consistency.

The characteristics of replication modes should be well considered prior to the implementation of a replication environment.

Lazy Mode

Lazy mode, which is the default mode of ALTIBASE HDB replication, provides the highest performance, but it may compromise from data integrity.

Lazy mode is asynchronous, meaning that the local server does not wait until a remote server is done applying a transaction. In Lazy mode, HDB service transactions and replication transactions are handled separately.

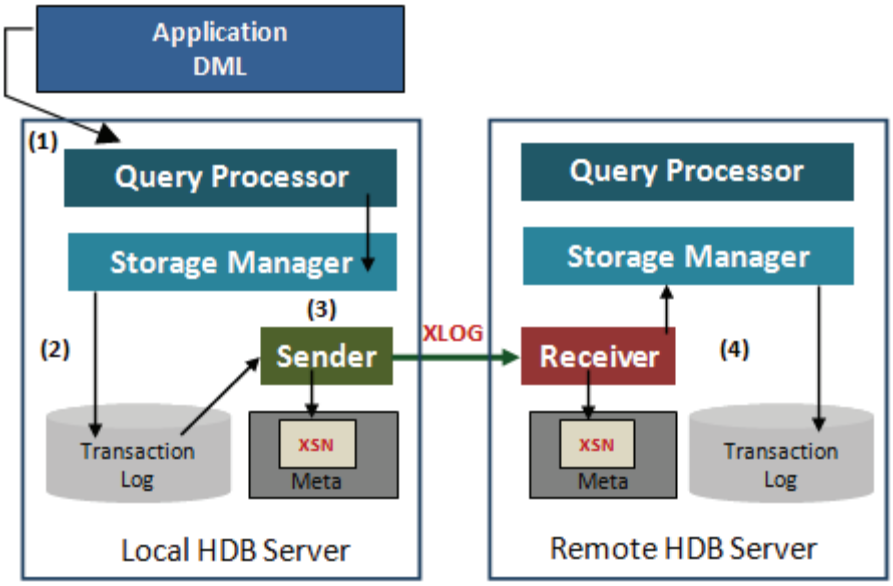

Figure 2: Altibase HDB Replication Lazy mode task flow

Lazy replication mode allows both locals and remote transactions to commit and run a conflict resolution during resynchronization. The resolution of such a conflict may be based on several conflict resolution methods provided by ALTIBASE HDB.

Below is the normal flow of tasks performed in Lazy mode as illustrated in Figure 2:
(1) An application transaction occurs on a local server (master transaction), and a DML statement is executed on a table that is a target for replication.

(2) ALTIBASE HDB applies the changes to the data from the master transaction and creates the transaction log records.

(3) The sender thread collects transaction logs recorded by the master transaction, converts them into XLOGs, records the XSN in the replication metadata table, and sends XLOGs out to the receiver thread on the remote server. On the local server, ALTIBASE HDB commits the master transaction, and continues processing other transactions without waiting for any information related to the replication.

(4) The receiver thread receives the XLOGs and commits the replicated transactions to its database instance on the remote server.

Because HDB service transactions and replicated transactions take place in complete isolation from one another, transactions do not influence each other, thus there is not a significant overhead on a local server from the replication process.

In fact, while other database solutions may introduce performance overheads up to 40% with their replication services, ALTIBASE HDB replication overhead is well within 10%, remaining near standalone performance.

Eager Mode

Data is replicated at multiple nodes for both performance and availability. While Lazy mode focuses on high-performance, Eager mode puts the focus on data consistency.

Although Eager mode is typically slower than Lazy mode, ALTIBASE HDB replication minimizes such performance differences with its parallel sender/receiver thread architecture.

Eager mode keeps all replication targets synchronized by updating all nodes as part of a single atomic transaction.

In Eager mode, when a master transaction occurs on a local server, the local server commits the transaction only after it has received confirmation that each of the change logs has been properly applied on the remote server(s).

In this synchronized model, even if the master transaction is successfully executed on the local server, if there is a replication conflict on the remote server, it is not possible to commit the master transaction on the local server.

In such cases, users must explicitly roll back the transaction in order to be able to execute the next transaction.

When a local server is internally required to commit transactions, for example when running in auto-commit mode, master transactions are automatically rolled back. As a result, both master transactions in conflict and replicated transactions are rolled back, which prevents data incon-

sistencies due to replication.

One benefit of Eager mode is that it is possible to replicate transactions in parallel, because they are synchronized. Therefore, when replication is running in Eager mode, multiple Sender and Receiver threads can operate in parallel. The level of parallelism is user configurable.

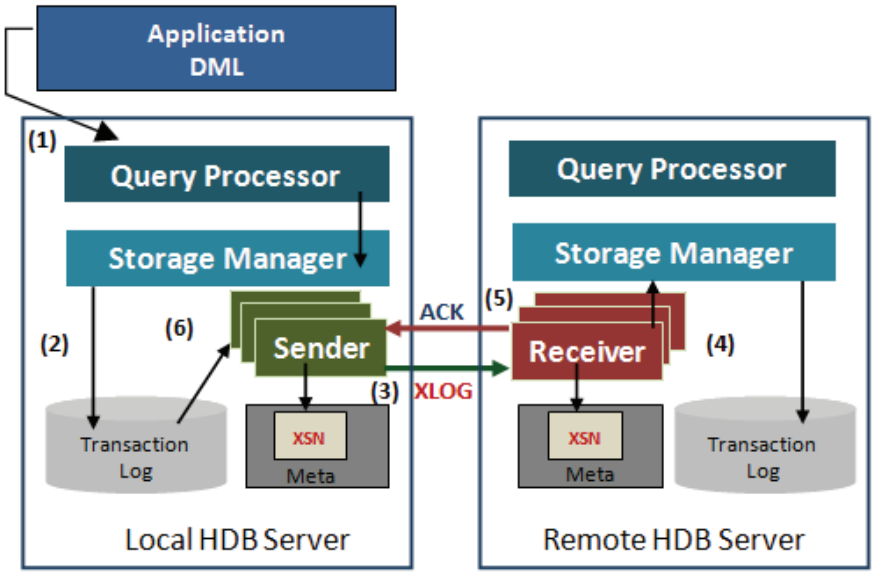

Figure 3: Altibase HDB Replication Eager mode task flow

Below is the normal flow of tasks performed in Eager mode replication as illustrated in Figure 3:
(1) An application transaction occurs on a local server (master transaction), and a DML statement is executed on a table that is a target for replication.

(2) ALTIBASE HDB applies the changes to the data from the master transaction, creates the transaction log records and blocks the master transaction.

(3) The sender thread collects transaction logs recorded by the master transaction, converts them into XLOGs, records the XSN in the replication metadata table, and sends XLOGs out to the receiver thread on the remote server.

(4) The receiver thread receives the XLOGs, creates its local transaction log records and commits the replicated transactions to its database instance on the remote server.

(5) The receiver thread sends ACK (the acknowledgment information) message back to the sender thread confirming that replicated transactions are committed on the remote server.

(6) Upon receiving ACK message, ALTIBASE HDB commits the master transaction on the local server.

Eager mode guarantees 100% data integrity. Eager mode is recommended if high availability of a database is the main focus, and planning an Active-Active architecture implementation.

ALTIBASE HDB Non Stop Services

Unexpected failures often occur in IT environments. Appropriate response for failures is just as important as replication strategies.

The problems that typically affect replication are:
- Communication failure between local and remote servers.
- Network failure.
- Abnormal shutdown of local or remote servers.

The following sections provide an overview of different failure scenarios and the recovery procedures.

Communication Failure between Local and Remote Servers

This is a scenario when the communication link between a local server (master server) and a remote server is lost. In this scenario, the following procedures take place.

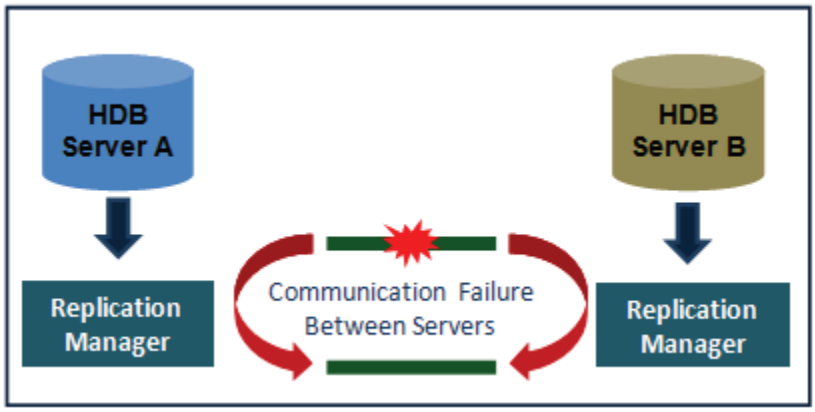

Figure 5: Communication Failure Scenario

Communication failure between servers:
1. Receiver threads on Server A and B roll back and terminate uncommitted transactions.
2. Sender threads on Server A and B record XSNs (XLOG Sequence Number) and attempt to connect to corresponding servers at intervals of 60 seconds.

Connection recovery:
1. Sender threads on Server A and B wake up Receiver threads on corresponding servers and perform replication by transmitting all XLOGs.
2. Receiver threads on Server A and B are created in response to connection requests from Sender threads on corresponding servers, and perform replication.
In such situations, it is also possible to take advantage of ALTIBASE HDB replication's ability to support multiple physical addresses at each node.

Network Failure

This is a scenario when the primary network communication between a local server and a database client application is lost. In this scenario, the following procedures take place.

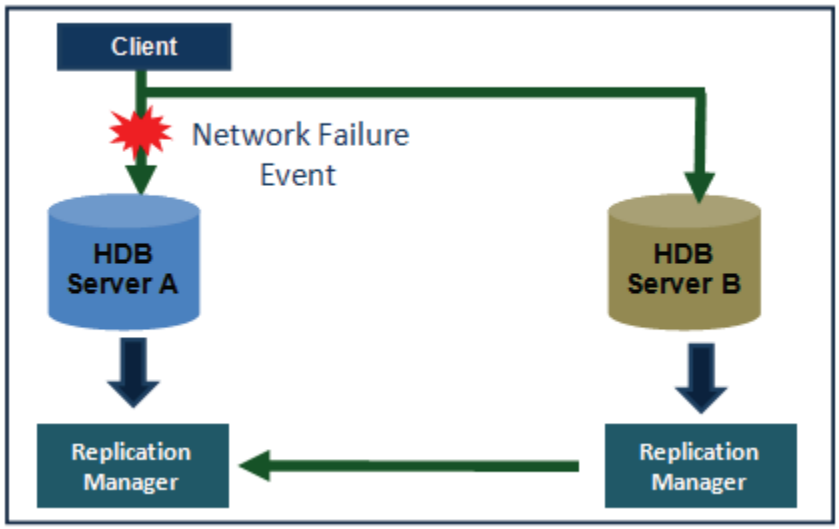

Figure 6: Network Failure Scenario

Primary network line failure:
1. Service is provided from Server B using a backup line.
Primary line recovery:
1. Service is provided from Server A again after the primary line is restored
2. Even while the primary line is down, Server B can continue sending change logs to server A.

Abnormal Shutdown of Local or Remote Server

This is a scenario when either a local server or a remote server goes out of service unexpectedly. In this scenario, the following procedures take place.

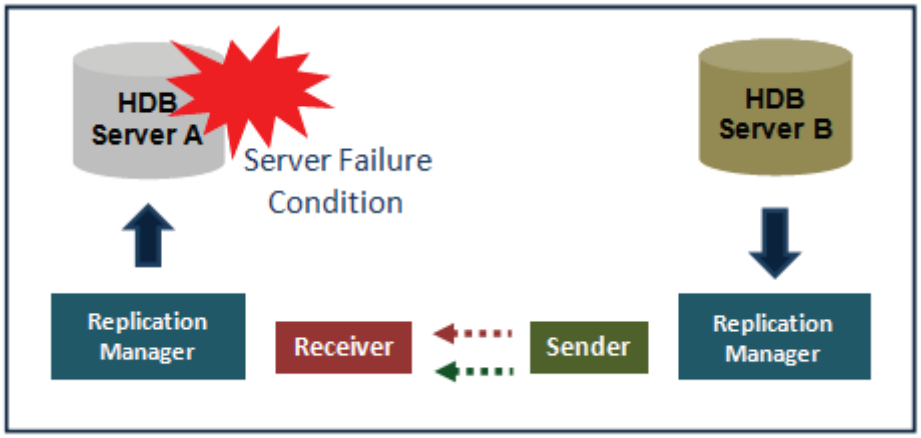

Figure 7: Abnormal Shutdown Case

Abnormal shutdown of Server A:
1. Receiver thread on Server B terminates, and Sender thread on Server B attempts to connect to Server A at user-defined intervals (e.g. every 60 seconds).

Recovery of Server A:
1. Sender thread on Server A automatically starts and performs replication with Server B.
2. Receiver thread on Server A is started by Sender thread on Server B, and performs replication.
3. Receiver thread on Server B starts replication after being started by the Sender thread on Server A.

ALTIBASE HDB Replication Conflicts

Data Conflict

The term "Data Conflict" refers to a case where records having the same primary key or records with constraints from replicated tables are changed by local transactions.

There are 3 types of data conflicts;
INSERT Conflict: This conflict occurs when a row with the same primary key value or a row with Unique constraint was inserted in more than one node within the replication environment.
UPDATE Conflict: This conflict may occur if current values of target column and before image in XLOG received are not the same when updating data.

ALTIBASE HDB replication sends both Before Image and After Image values when an update transaction is processed. If Before Image that is sent to a peer server does not match Current Image on the peer server, an update conflict occurs.

By default, when there is an update conflict, ALTIBASE HDB does not apply the After Image value on the peer server, but creates a log entry for the error condition instead.

To override this behavior to apply an After Image value, users can set REPLICATION_UPDATE_REPLACE property value to 1. This also skips the creation of an error log entry step.

An example scenario for an Update Conflict is shown below:

	Server A	Server B
Current value	C2=10	C2=10
Update transaction on the same primary key record on both Server A and B at the same time.	C2=30 (10→30 update)	C2=40 (10→40 update)
Value after the transaction	Sender: XLOG (10 → 30)	Sender: XLOG (10 → 40)

Values on both server A and B were the same initially, but UPDATE transactions that occurred at the same time have changed them into different values.

In this case, server A sends XLOG (10→30) to server B, while server B sends XLOG (10→40) to server A. However, neither XLOG can be applied since Before Image is different from Current Image on both servers. Therefore, both servers end up with different values.

DELETE Conflict: This conflict occurs when two transactions originate from different applications, with one transaction deleting a row that the other transaction updates or deletes.

Replication Gap

Another issue in replication environments operating in an asynchronous mode is the possible delays in the data transfer process. When the speed of creating or recoding transaction logs on a sender node is faster than the speed of applying XLOGs by a receiver thread on a receiving node, sending and receiving nodes can potentially end up with different data at a specific point in time.

In other words, if a remote server cannot keep up with the speed of a local server, the remote server ends up with stale or inconsistent data. This condition is known as a replication gap. ALTIBASE HDB replication provides tools for monitoring such conditions and enables users to avoid them through effective system configuration guidelines.

ALTIBASE HDB Replication Conflict Resolution

To help users deal with replication conflicts discussed in the previous section, ALTIBASE HDB replication provides the following conflict resolution schemes:

- User-Oriented scheme
- Master-Slave scheme
- Timestamp-Based scheme

User-Oriented Scheme

This scheme allows users to determine the policies for handling data conflicts on a case-by-

case basis.

INSERT Conflict:
ALTIBASE HDB replication synchronizes data based on primary keys. Therefore, it does not allow updating the primary key of a replicated table. If a primary key must be updated, then a DELETE/INSERT approach is recommended (delete existing data then insert new data).

When an insert conflict occurs, the INSERT statement fails, and a conflict error message is recorded in the altibase_rp.log file. In order to resolve this conflict, ALTIBASE HDB provides users with the REPLICATION_INSERT_REPLACE property. Using this property, users can establish their own policy for handling such conflicts.

This property specifies whether to keep inserted contents if an insert conflict occurs during replication. If its value has been set to 0, the insert will not be committed, and the data conflict will be handled as an error, whereas if its value has been set to 1, the data conflict will be ignored and the insert will be committed. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB replication is running.

REPLICATION_INSERT_REPLACE=1: DELETE then INSERT
REPLICATION_INSERT_REPLACE=0: Report the conflict condition in the logs.

UPDATE Conflict:
When an update conflict occurs, the UPDATE statement fails, and a conflict error message is recorded in the altibase_rp.log file. In order to resolve this conflict, ALTIBASE HDB provides users with the REPLICATION_UPDATE_REPLACE property. Using this property, users can establish their own policy for handling such conflicts.

This property specifies whether to keep the updated contents if an update conflict occurs during replication. If this value has been set to 0, the update will not be committed, and the data conflict will be handled as an error, whereas if this value has been set to 1, the data conflict will be ignored and the update will be committed. This property can be changed using the ALTER SYSTEM statement while ALTIBASE HDB replication is running.

REPLICATION_UPDATE_REPLACE=1: Update
REPLICATION_UPDATE_REPLACE=0: Report the conflict in the logs.
DELETE Conflict:

If a delete conflict occurs, the DELETE statement fails, and a conflict error message is recorded in the altibase_rp.log file. In this case, the conflict is only reported, and ALTIBASE HDB does not provide a property for establishing a user policy.

Users can also minimize the occurrence of replication data conflicts through different design approaches.

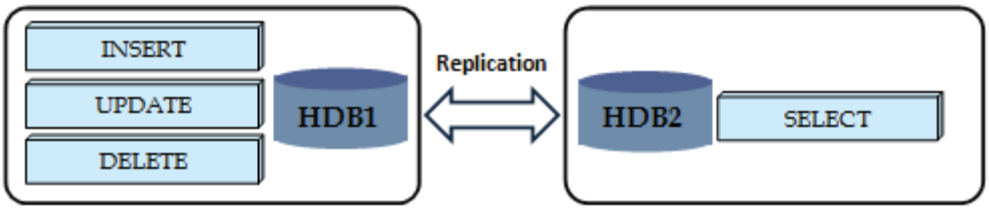

Figure 8: DML and SELECT Statement Distribution

In one approach, DML and SELECT operations can be split across participating nodes in replication. As illustrated in the Figure 8, while ALTIBASE HDB1 instance processes INSERT, DELETE and UPDATE statements, ALTIBASE HDB2 instance processes SELECT transactions only.

Another effective design involves establishing different primary key ranges between participating nodes in the replication environment. As illustrated in Figure 9, certain primary key ranges can be set for ALTIBASE HDB1 and others for ALTIBASE HDB2. Transactions related to the primary key on ALTIBASE HDB1 instance are processed on ALTIBASE HDB1 while others are processed on ALTIBASE HDB2 database.

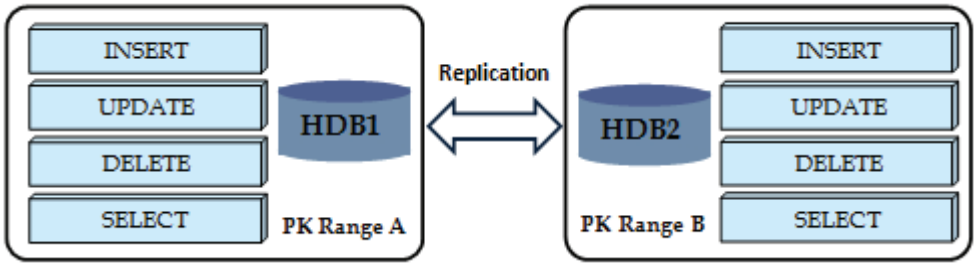

Figure 9: Use of primary key ranges

Master-Slave Scheme

Another approach to dealing with replication conflicts is to distinguish between master and slave nodes in a replication environment.

When creating a replication environment, by specifying a master or a slave designator with the CREATE REPLICATION statement, a server to be replicated can be distinguished by its specific role in replication.

The CONFLICT_RESOLUTION field in the SYS_REPLICATIONS_ metadata table identifies a server's role as follow;
0 = Not specified
1 = Master
2 = Slave

Based on these designators, replication nodes can handle replication conflicts differently. The table below shows how a master and a slave deal with conflict resolutions.

Node Role	Conflict Resolution Scheme	
Master	INSERT	Ignores INSERT operation.
	DELETE	Ignores DELETE operation.
	UPDATE	Ignores UPDATE operation.
Slave	INSERT	Inserts new record after deleting an existing record.
	DELETE	Ignores DELETE operation.
	UPDATE	Updates the record.

Operating as a master:
- INSERT conflict: Not committed.
- UPDATE conflict: Not committed.
- DELETE conflict: Not committed.
- XLOG transferred from a slave is processed as usual.

Operating as a slave:
- INSERT conflict: If an insert conflict occurs because an attempt was made to insert data having the same primary key as an existing record, the existing record is deleted and the new record is added.

If an insert conflict occurs for any other reason, the INSERT statement fails, and a conflict error message is recorded in the altibase_rp.log file.
- UPDATE conflict: If an update conflict occurs because an attempt was made to update a record having a value different from the Before Image value on another database server, from which data for replication is being propagated, the conflict is ignored, and UPDATE statement succeeds despite the conflict.

If an update conflict occurs for any other reason, the UPDATE statement fails, and a conflict error message is recorded in the altibase_rp.log.
- DELETE conflict: If a delete conflict occurs because no record having with the specified primary key exists, the DELETE statement fails, and a conflict error message is not recorded in the altibase_rp.log. If a delete conflict occurs for any other reason, the DELETE statement fails, and a conflict error message is recorded in the altibase_rp.log.
- XLOG transferred from master is processed as usual.

Timestamp-Based Scheme

ALTIBASE HDB replication provides a timestamp-based scheme to ensure that both servers have the same data in an Active-Active replication environment.

When a data conflict occurs during an Active-Active replication, if the value of the REPLICATION_TIMESTAMP_RESOLUTION property is set to 1 and a TIMESTAMP column exists on the target table, then a timestamp-based resolution scheme is used to resolve the conflict.

Conversely, even if a TIMESTAMP column exists on a replication target table, if the value of the

REPLICATION_TIMESTAMP_RESOLUTION property has been set to 0, only a normal conflict resolution scheme is used.

ALTIBASE HDB supports timestamp-based scheme only for INSERT and UPDATE operations. Below are conflict resolution handling scenarios based on the type of conflict:

INSERT Conflict
1. If the data to be inserted has the same primary key as existing data, the timestamp value of After Image of the data is compared with that of the existing data.
2. If the TIMESTAMP value of After Image of the data is equal to or greater (more recent) than the TIMESTAMP value of the existing data, existing data is deleted, and the value of After Image of the data is added as new data.

UPDATE Conflict
1. The TIMESTAMP value of After Image of the data is compared with that of the data to be updated.
2. If the TIMESTAMP value of After Image of data is equal to or greater (more recent) than the existing data, the data is updated with After Image value.
3. When an UPDATE is performed, the TIMESTAMP value of After Image of the data is maintained. In other words, independent system time values are not used.

The following restrictions apply when using the timestamp-based scheme:
- Every table to be replicated must contain a TIMESTAMP column.
- REPLICATION_TIMESTAMP_RESOLUTION property must be set to 1.

Because ALTIBASE HDB supports a timestamp-based scheme based on tables, even if a replication target table has a TIMESTAMP column, if the value of REPLICATION_TIMESTAMP_RESOLUTION property for that table has been set to 0, a normal conflict resolution scheme will be used.

ALTIBASE HDB Replication Conflict Resolution Flow

ALTIBASE HDB tablespace is a logical storage unit for storing tables, indexes and other database objects.

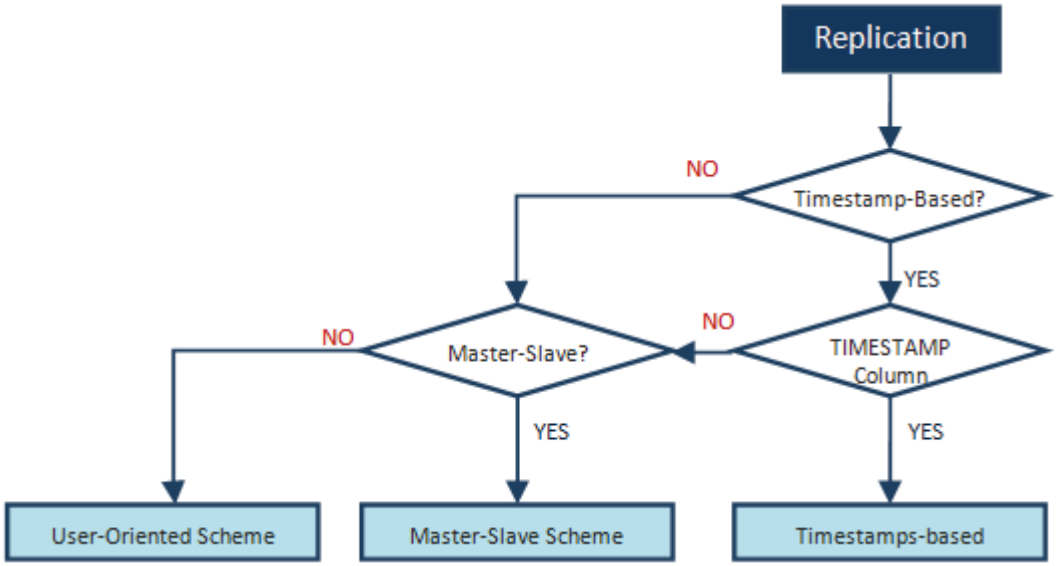

Figure 10: Altibase HDB Conflict Resolution Flow Logic

ALTIBASE HDB Replication Extra Features

Offline Replication

In an Active-Standby replication environment, when a server providing service (active server) has a failure, logs cannot be sent to the remote server (standby server).
Use of offline replication allows logs that could not be sent to a standby server to still be accessed and processed by that standby server.

There are a couple of different configuration options for implementing an offline replication environment;

- Allow active and standby servers to share a disk (NFS or shared disk) for their transaction log files.
- Implement an FTP service, where nodes can access the transaction logs of each other over File Transfer Protocol (FTP).

In order to initiate an offline replication operation, the OPTIONS property in the SYS_REPLICATIONS metadata table should be set to 2.

The log path of an active server can be specified in the Log_dir attribute of the ALTER REPLICATION statement.

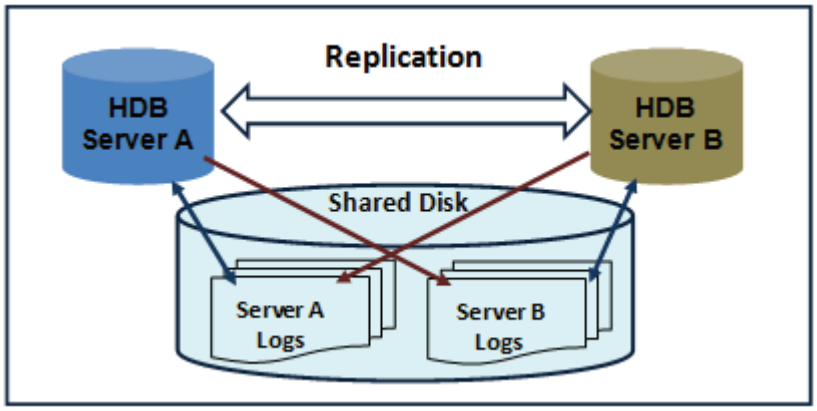

Figure 11: ALTIBASE HDB Offline Replication

N-Way Replication Connections

ALTIBASE HDB replication supports n-way connections up to 32 connections. However, XLOGs received from a sender node cannot be transferred again to another node. Therefore, when configuring 3 or more nodes in a replication environment, users should create replication objects in pairs as illustrated in Figure 12.

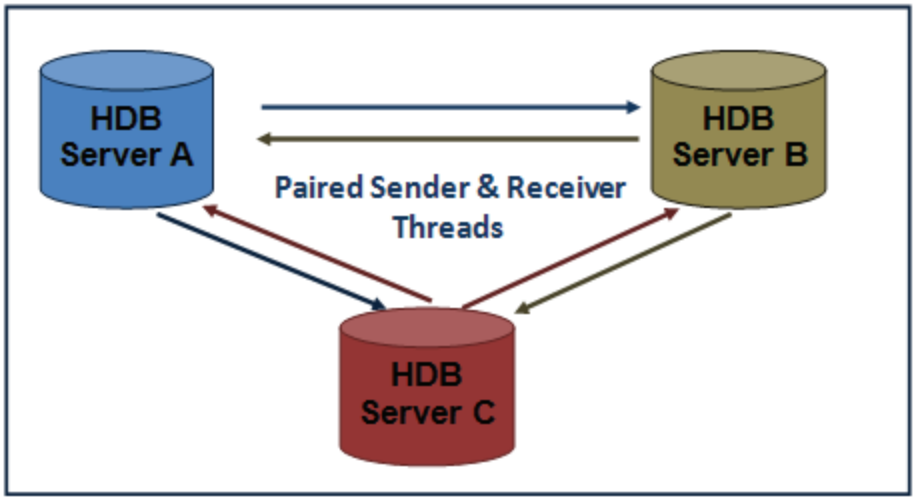

Figure 12: ALTIBASE HDB N-Way Replication Connections

Executing DDL Statements in Replication

Execution of DDL statements is permissible in ALTIBASE HDB replication, however some restrictions apply.

ALTIBASE HDB provides the REPLICATION_DDL_ENABLE property to permit execution of DDL statements in replication. If this property's value is set to 1, the following DDL statements can be executed:

- ALTER TABLE table_name ADD COLUMN
- ALTER TABLE table_name DROP COLUMN
- ALTER TABLE table_name ALTER COLUMN column_name SET DEFAULT
- ALTER TABLE table_name ALTER COLUMN column_name DROP DEFAULT
- ALTER TABLE table_name TRUNCATE PARTITION TRUNCATE TABLE
- CREATE INDEX
- DROP INDEX

It is still possible to execute DDL statements that are not included on the list above by pausing or stopping the replication service first, dropping subject tables, executing DDL, re-registering the objects and finally resuming the replication service normally.

Replication in a Multiple IP Network Environment

In ALTIBASE HDB, replication is supported in a multiple IP network environment. In other words, it is possible to perform replication between two hosts having more than two physical network connections.

In order to ensure high system performance and quickly overcome communication failures, systems can have multiple physical IP addresses assigned to them when a replication object is created.

In such an environment, a sender thread uses the first IP address available to access remote servers and performs replication tasks when replication starts. But if a problem occurs while this task is underway, the sender thread stops using this initial connection, switches to an alternate IP address, and tries again.

Audit Utility

ALTIBASE HDB provides an Audit utility for monitoring and managing replication status and data conflicts.

The Audit utility compares an ALTIBASE HDB database with either another ALTIBASE HDB database or an Oracle database (serving as the remote database) on a table-by-table basis, and outputs information about any data conflicts it may discover.

It also includes a function to synchronize two databases in the event a data conflict is discovered.

Audit utility can automatically identify the following data conflicts;
- When a record based on a primary key can be found in a master database but not in a slave (MOSX inconsistency).
- When a record based on a primary key can be found in both master and slave databases but contents are different (MOSO inconsistency).
- When a record based on a primary key can be found in a slave database but not in a master (MXSO inconsistency).

In replication, a synchronization policy is defined as a policy that specifies how to synchronize inconsistent records. ALTIBASE HDB Audit utility provides four synchronization policies.
- SU Policy: It resolves MOSO inconsistencies by updating a slave database with contents of a master database.
- SI Policy: It resolves MOSX inconsistencies by inserting records from a master database into a slave database.
- MI Policy: It resolves MXSO inconsistencies by inserting records from a slave database to a master database.
- SD Policy: It resolves MXSO inconsistencies by deleting records from a slave database.

MI and SD policies are mutually exclusive, meaning that they cannot both be set at the same time.

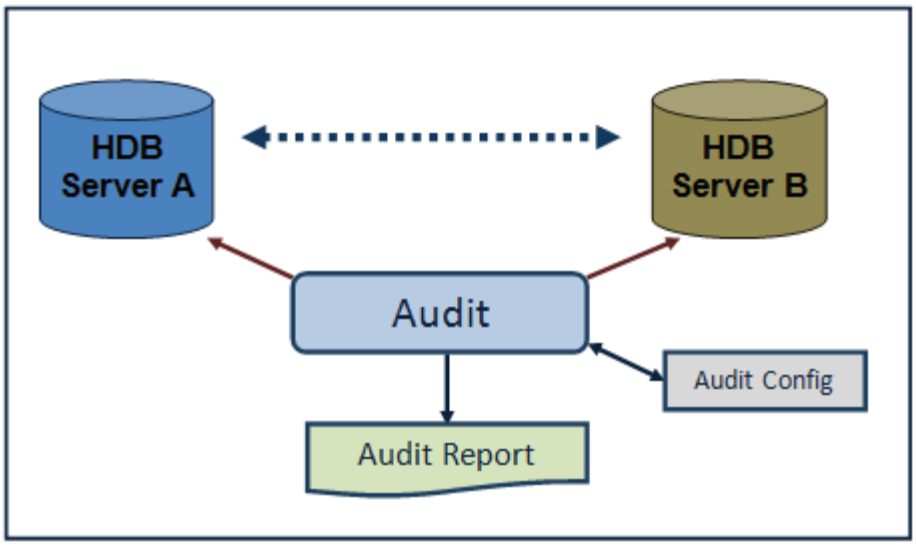

Figure 13: ALTIBASE HDB Audit Utility Architecture

An Audit configuration file is used for setting all options of Audit utility's execution environment. This file contains connection information, Audit function settings, and synchronization policies.

The Audit utility can perform two types of operations based on configuration settings defined in the Audit configuration file.

Diff operation creates an audit report file that contains inconsistencies identified by Audit utility.

Sync operation first identifies inconsistencies between a master and a slave database, and then bi-directionally resolves them according to the policies set in the configuration file.

ALTIBASE HDB Replication Considerations

- Replicated tables must have primary keys.
- Primary keys of replicated tables must not be updated.
- Table descriptions (including primary key and not null column information) in both local and remote servers should be the same.
- Because of possibility of data conflict due to a replication gap, replicated tables are not recommended to use a trigger or a foreign Key.

- DDL statements cannot be executed on tables for which Replication Recovery option has been specified.
- DDL statements cannot be executed while replication runs in Eager Mode.
- When permitted DDL statements are executed, those subject tables are locked. If a sender thread transfers a replication log at that time, the receiver thread won't be able to properly implement the log's changes.
- Sequence objects cannot be included in a replication list. If users require the use of sequence objects, they should create a table to generate sequential numbers and use that table just like a sequence object in replication.
- LOB type column cannot be either a primary key or a unique key.
- There is no restriction on the size of XLOG for memory tables. However, for disk tables, the size of XLOG for each row cannot exceed 128KB.
- When replicating partitioned tables, the partition method should be the same on both local and remote servers.
- The maximum number of servers participating in replication is 32.
- Character and National Sets should the same on both local and remote servers.
- The performance of replication can be low if a replication object contains both memory and disk tables. The performance of a disk table is typically much lower than a memory table. Therefore, it is recommended to separate memory and disk tables for higher performance.
- Gigabit Ethernet is highly recommended for higher replication performance.
- Conditional clause cannot be used while replication is running in Eager mode.
- When using Conditional Clause, it is preferable to use a primary key column as a condition column. When using an ordinary column as a condition column, it is necessary to ensure that data in the conditional column is not changed in order to prevent log-related overhead.
- When using Conditional clause it is also necessary to ensure that data in the conditional column is not changed when performing replication in an Active-Active mode.
- With ALTIBASE HDB replication, it is possible to have local and remote servers in different geographical locations; however, the replication performance may decrease in accordance with network bandwidth and latency.
- Replication is permissible between heterogeneous servers, however if the byte order is different, the necessary operation for changing the byte order may potentially impact overall replication performance.

Company Overview

ALTIBASE is a leading provider of data performance solutions that provide real-time access, analysis, and distribution of high volumes of data in mission critical environments.

As the importance of real-time information grows, ALTIBASE data performance solutions deliver real-time access to your data whether your data is at-rest in a database or it is still in-motion across the wire.

ALTIBASE helps its customers maximize their data investments by providing real-time data performance solutions. In today's competitive business environment, ALTIBASE enables companies to drastically improve the speed of data access and analysis across the enterprise.

ALTIBASE delivers its data performance solutions through two key products:

• ALTIBASE® HDB™ is a hybrid relational DBMS that combines in-memory and on-disk storage in a single relational database.

• ALTIBASE® DSM™ is a data event middleware that filters, analyzes and distributes high-volume data streams in real time.

ALTIBASE

Copyright 2011 by Altibase Corporation.
All Rights Reserved.