Altibase Technical Documentation | Application Development

Altibase Development Guide

Copyright ⓒ 2000~2026 Altibase Corp. All Rights Reserved.

This document is provided by Altibase for informational purposes only, and its contents are subject to change without prior notice. This document may contain errors, and Altibase makes no express or implied warranties of merchantability or fitness for a particular purpose.
Altibase develops and announces the features and functionality described in this document at its own discretion. Altibase may also hold patents, trademarks, copyrights, and other intellectual property rights related to this document.

Altibase Corp.

10F, Daerung Post Tower II, 306 Digital-ro, Guro-gu, Seoul 08378, Republic of Korea

Tel	+82-2-2082-1114
Fax	+82-2-2082-1099
Customer Support Portal	https://support.altibase.com/en/
Website	https://altibase.com/en/

Contents

01 Overview . . . 4

02 Considerations when Designing . . . 5

Considerations for replication environment . . . 5
Consideration for high availability (HA) and backup . . . 6
Considerations for designing table . . . 7
Considerations for Using Partition Tables . . . 8
Considerations for Hardware preparation . . . 8
Not Support Feature . . . 9

03 Considerations when Developing . . . 10

Choosing DBMS connection type . . . 10
Thread program, connection pool management . . . 10
Checkpoints when disconnecting an application . . . 11
Session Timeout . . . 12
Considerations when using SELECT Cursor . . . 13
Checking SQL Error . . . 14
Considerations for LOB type . . . 14
Using Prepared Statement . . . 15
SQL Tuning . . . 15
Cautions during the bulk change operation . . . 16

04 Altibase Trace Logs . . . 18

Classification of trace logs . . . 18

05 CLIENT APPLICATION Error Messages . . . 25

Connection does not exist . . . 25
Communication link failure . . . 25
Calculate stack overflow . . . 25
Conversion not applicable . . . 26
Invalid cursor state . . . 26
Not defined cursor . . . 26
Invalid request to process the SQL statement . . . 26

© Altibase Corp.

Invalid literal . . . 27
Invalid length of data type . . . 27
Indicator variable required but not supplied error . . . 27
Invalid size of data to bind to host variable Data Size … . . . 27
Too many pages are allocated . . . 28
The Tablespace does not have enough free space . . . 28
The transaction exceeds lock timeout specified by user . . . 28
The update log size '…' is bigger than TRX_UPDATE_MAX_LOGSIZE '…' . . . 28
String data right truncated . . . 29
Value overflow . . . 29
Several statement still opened . . . 29

© Altibase Corp.

01

Overview

This technical document describes things to consider in the development environment when using ALTIBASE DBMS.

For reference documents for each specific development environment, please refer to the technical documents prepared for each development environment.

General information is described in this document, and ALTIBASE error messages are described later in the document.

Please also refer to the following documents according to your environment.

1. "Configuration Guide For Minimizing Disk I/O Contention"
2. "Considerations for Altibase Backup Policy"
3. "Altibase Replication Configuration Guide"
4. "Altibase Replication Constraints Guide"
5. "System Data Capacity Estimation Guide for Altibase Operations"
6. "Altibase Precompiler (APRE*C/C++) Guide"
7. "Performance Tuning Guide"
8. "JAVA Developer's Guide"
9. "ORACLE to ALTIBASE Conversion Guide"

Info

For errors and improvements related to this document, please contact the technical support portal or technical support center.

- Technical support portal: http://support.altibase.com/en
- Technical support center: +82-2-2082-1114

Info

This document is provided for informational purposes only and is subject to change without prior notice. It may contain errors, and Altibase makes no express or implied warranties of any kind, including but not limited to warranties of merchantability or fitness for a particular purpose.

The development, release, and timing of any features or functionality described for Altibase products in this document are at the sole discretion of Altibase.

Altibase may hold patents, trademarks, copyrights, or other intellectual property rights related to the contents of this document.

© Altibase Corp.

02

Considerations when Designing

This chapter describes things to consider when designing a DB or system using ALTIBASE. It is recommended to consider the following items and refer to the design because not only the impact on the design stage but also on the development stage and the operation stage is significant.

Considerations for replication environment

① ALTIBASE network-based replication method

ALTIBASE adopts a replication method of synchronizing data by sending/receiving change transaction logs based on the network. In other words, it is not a usual way to share disk that users are commonly aware of. Therefore, it is necessary to design in consideration of the synchronization part due to data transmission delay, which must occur in a network environment.

There are two replication methods: The lazy method and the Eager method. The Lazy method allows the delay of data transmission, and on the other hand, the Eager method does not allow it. Therefore, the Lazy method is adopted for tasks that allow data transmission delay, and the Eager method is set and operated appropriately for each task if data does not necessarily need to be synchronized.

Comparison	Lazy Method	Eager Method
Performance	90% performance level of a single server	Performance degradation compared to the Lazy method
Synchronization	This is a structure that checks only the success of transmission of the changed transaction log to another server, so it is a high performance, but synchronization may be delayed.	To prevent data inconsistency on another server, performance is rapidly delayed because it is also committed locally on another server, but the synchronization is not delayed.

② Consider measures to prevent data inconsistencies

In a dualized environment, if records with the same primary key are modified simultaneously with different values on both servers, unwanted data inconsistency may occur due to server conflicts. To prevent this, the following design considerations are required:

1. Business Logic Separation

   Modification operations (Insert/Update/Delete) must be executed only on one server within the group, while the other server is used exclusively for read operations.
   For example, assigning server A as read-only for real-time queries and server B as write-only for modification tasks can fundamentally reduce the risk of conflicts.

2. Data Responsibility Separation by Server

© Altibase Corp.

To avoid simultaneous updates to the same PK, divide and manage the PK range by server. Example: Design so that odd-numbered member IDs can only be modified on server A, and even-numbered IDs only on server B.

③ Log-based replication method

Altibase replication is a form sending changed log of a transaction occurring on the local server. This means that even if one million statements are changed with one SQL statement, the same is the structure in which a million changed transaction logs are transmitted, not the same for one SQL statement transmitted to a remote server.

In addition, it is important to refer to the ALTIBASE technical documents "Altibase Replication Configuration Guide" and "Altibase Replication Constraints Guide" for major considerations of the replication environment.

Consideration for high availability (HA) and backup

Under normal circumstances, when a failure occurs and recovery is performed, replication automatically synchronizes data by sending a transaction log to the other side that has not yet sent to the point of failure. However, due to the nature of the network, it is necessary to consider whether there is a task that cannot perform a fail-over service immediately due to data inconsistency caused by the inability to transmit the change transaction log at the time of failure.

If the service is available only after the data is synchronized, ALTIBASE provides an offline replicator to reflect the undeferred data. However, there is a limitation that it can be used only when the disk of the failed server is accessible. (In order to use this function, it is recommended to build a shared disk device.)

If it is not an environment where the off-line replicator can be used, it is necessary to establish a HA policy that accumulates raw-data at the point of failure in the program and reflects it on the normal server as much as necessary. For example, in the case of securities companies, all data going to the world is managed by unique numbers. Since these numbers are the same in the external system and the DBs, in some cases, a business logic that reflects only the portion of the difference between the external system and the DB system in the normal server is used.

In addition to HA, it is necessary to establish a backup policy in preparation for failures caused by data file damage due to physical disk failure or transaction log file corruption.

The Altibase backup method provides both full and incremental backup methods using the archive mode.

Therefore, users can choose one of the following backup methods.

1. Cold backup of all necessary files while offline
2. Backup method using iLoader while online
3. Full backup of the database or tablespace unit while online
4. Incremental backup of the database or tablespace unit while online

© Altibase Corp.

The following describes each backup method.

Backup Method	Description
iLoader Backup	Only the current values of records are stored for each major table, so only the data at the time of backup by iLoader is backed up.
Cold Backup	After the ALTIBASE process is shut down, all necessary files are copied and stored, so recovery is possible only up to the point when the files were copied.
DB Full Backup	Because the entire database image, including the transaction log files, is backed up at the time of backup, recovery is possible up to the point where there is no damage to the files.
DB Incremental Backup	A full backup is performed once at the beginning, and only the changed image is backed up from then on. Since the transaction log files are also backed up, recovery is possible up to the point where there is no damage to the files. The more incremental backups there are, the longer recovery takes, so this needs to be considered.

For the backup, please refer to "Considerations for Altibase Backup Policy".

Considerations for designing table

① Select the storage location of the table according to the purpose

ALTIBASE supports both memory and disk tablespaces.

1. Memory tablespace Suitable for tasks that require very high-speed processing performance. Recommended for tables with high transaction frequency and short data validity periods (e.g., session information, same-day transaction data, etc.).
2. Disk tablespace Suitable for tasks that require long-term storage and large-volume data loading. Mainly used for historical data and tables focused on large-scale queries.
3. Using a Hybrid Partitioned Table (HPT) ALTIBASE provides the HPT feature, which allows a single logical table to be configured with both memory and disk partitions at the same time. Example: The most recent N days of data can be stored in a memory partition while older data is stored in a disk partition, allowing it to be managed as a single table. From the application's perspective, there is no need to distinguish between separate tables, so it can be queried as a single table, which improves manageability. However, even when using a Hybrid Partitioned Table, the performance of queries that include the disk partition is still affected by disk I/O.

② Considerations for Column type

When selecting a column type in ALTIBASE, storage space efficiency, processing performance, and operational characteristics must be comprehensively considered. In particular, for numeric and date types, the difference between native and non-native types and the cost of internal conversion can directly affect performance.

The following describes considerations when selecting a column type.

© Altibase Corp.

Consideration	Description

Numeric Type	Altibase's numeric types are broadly classified into native types and non-native types.
	- Native Types (Better performance, no internal conversion required)
	  - `SMALLINT` (2 Bytes)
	  - `INTEGER` (4 Bytes)
	  - `BIGINT` (8 Bytes)
	  - `DOUBLE` (8 Bytes, floating-point)
	- Non-Native Type (Internal conversion occurs during storage, lower performance)
	  - `NUMERIC(p, s)` (Supports up to 38 digits with defined precision `p` and scale `s`)
	
	1. Prefer Native Types
	If the same data can be represented, choosing a native type reduces conversion cost and improves performance. Example: For simple integer data, use `SMALLINT`, `INTEGER`, or `BIGINT` based on size instead of `NUMERIC`.
	2. Integer Data
	Choose the smallest appropriate type among `SMALLINT`, `INTEGER`, or `BIGINT` based on the value range to save storage.
	3. Floating-Point Data
	Use `DOUBLE` if fixed precision is not strictly required, as it offers better performance. Use `NUMERIC(p, s)` only when exact precision must be guaranteed.
	4. Consider Aggregation Operations
	For columns frequently used in `SUM` or `AVG`, `DOUBLE` or `BIGINT` is recommended. Using `NUMERIC` may incur additional conversion cost.

Date Type	If the date-related operation is important, select the Date type. If it is limited to search and comparison operations, select the Char/Varchar type. (Date type is fixed at 8 bytes)
Join	Columns used in JOIN conditions must be defined with the same data type. Comparisons between different data types trigger internal type conversion, which can lead to performance degradation.

③ In the case of replication, it must have a primary key

④ Foreign key is not used because it degrades transaction performance.

Considerations for Using Partition Tables

For the ALTIBASE partition table (as of version 7.3), partitioned local index and non-partitioned global index are supported, and partitioned global index is not supported. Therefore, it is recommended that the partition table uses historical data for the purpose of storage for each month and purposes of business because a significant performance decrease occurs when querying in the form that excludes the partition key in the condition clause that queries the entire partition table.

Considerations for Hardware preparation

The hardware-related considerations to the information above are summarized and explained in this section.

© Altibase Corp.

Consideration	Description
Replication	Because it uses a network, it is recommended that the connection is replicated between the servers by using a separate Giga-bit LAN card or using a private network.
Disk	When considering backup, in addition to the estimated DB capacity, disk space is required to store more archive log files along with the entire tablespace.

In addition, the policy of ALTIBASE for redo log files can be viewed as infinite. It is recommended to allocate enough space on the disk where the redo log files will be stored, as there are cases where it is inevitable to create a large number of redo log files due to bulk change operations because the policy is to create a new redo log file instead of reusing redo log files.

For more detailed information, please refer to the "System Data Capacity Estimation Guide for Altibase Operations".

Not Support Feature

ALTIBASE complies with the SQL92 standard, but there are some differences compared to other DBMSs.

For more detailed information, please refer to the technical documents below.

1. MSSQL to ALTIBASE Conversion Guide
2. ORACLE to ALTIBASE Conversion Guide
3. Altibase/Oracle Comparison

© Altibase Corp.

03

Considerations when Developing

This chapter describes considerations in the development stage are summarized and explained.

Choosing DBMS connection type

① Choosing ALTIBASE connection type

Connection Type	Description
CONNTYPE=1	TCP/IP connection type, normal connection method
CONNTYPE=2	UNIX Domain Socket connection type, only available in the local server
CONNTYPE=3	IPC connection type, only available in the local server

The CONNTYPE=n means the connection type, and when the connection string is specified in the program, the user can specify it explicitly. If DBMS and application are located on separate servers, only CONNTYPE=1 can be used. However, if DBMS and application run on the same server, it is recommended to select CONNTYPE=2, 3 to reduce the cost of communication.

② Auto-commit mode is the default setting for ALTIBASE

JDBC connects in auto-commit mode according to the standard specification. Other applications are determined by the "AUTO_COMMIT" property in the $ALTIBASE_HOME/conf/altibase.properties file by default unless the user changes. If this property value is "1", it operates in auto-commit mode. (Auto-commit means that the commit is automatically performed after the change transaction is completed normally)

Therefore, if the user wants to change this property value to Non auto-commit, ALTIBASE must be restarted after changing the property value to "0" in the configuration file. If the user is controlling by session, the user can execute the following SQL after connecting to the DBMS.

```
ALTER SESSION SET AUTOCOMMIT = FALSE; (In the case of Java, it is controlled using setAutoCommit function.)
```

(For the ALTER SYSTEM command, it can be changed in real time to reflect the entire system, but it will be restored after restarting, so it is recommended to modify and restart the property that affects the entire property through the property file.)

Thread program, connection pool management

In a thread environment program, if multiple threads share one connection, the developer must perform concurrency control for the connection. ALTIBASE exchanges the promised protocols during processing session-server communication. Generally, it goes through the process

© Altibase Corp.

of PREPARE→BIND→EXECUTE→FETCH, etc. In this situation, an error occurs in the process that the other protocol is interrupted and the session is incorrectly processed.

```
Communication link failure (EXEC->INVL)

Communication link failure (PREP->EXEC)

Invalid request to process the SQL statement
```

The above errors occur when one connection is executing another SQL in the absence of concurrency control while executing any SQL while using the thread program or connection pool. Therefore, when the above error occurs, first, it is necessary to check whether concurrency control or connection processing in the program written by the developer does not cause an error to interrupt.

(Occasionally, if the timer is set in the user program, it may occur even if it is not a concurrency control problem as described above. For example, another SQL statement is executed while the FETCH protocol is not yet completed because the timer operates during the SELECT~FETCH process. if it is set, the above error may occur.)

Checkpoints when disconnecting an application

The SQL statements executed in the program are prepared in individual programs and servers. It is recommended to develop code to release resources when these SQL statements are used. In particular, in the case of Java, memory increase may occur, so the following code is important.

```
Connection con = pool.getConnection();
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery();

try
{
   ….
} exception ()
{
   …..
}
finally
{
    // Release all resource used within the program.
    rs.close();
    stmt.close();
    conn.close();
}
```

In the case of development such as CLI/ODBC, SQL_DROP must be executed with the SQLFreeStmt function at the end of use after the memory is allocated with SQLAllocStmt. If it is reused, use the SQL_CLOSE option. Otherwise, if it is completely disabled, use the SQL_DROP option to explicitly disable it.

© Altibase Corp.

Session Timeout

A session is the connection of the program connected to the DB. Each session is managed by the timeout policy of ALTIBASE. ALTIBASE provides five timeout policies. (The color shown below can be controlled only in the connection string.)

Timeout Type	Session state after an error	Description
CONNECTION_TIMEOUT	Disconnected	Option to set the timeout in the blocked state between transmission/reception on the network while connected to the DB
TIMEOUT	Disconnected	Occurs when the time to try to connect to the DB exceeds this setting
QUERY_TIMEOUT	Connected	Occurs when the total time to execute a query while connected to the DB exceeds this setting
FETCH_TIMEOUT	Disconnected	Occurs when the time interval during which the FETCH protocol occurs exceeds this value after executing the SELECT statement while connected to the DB
IDLE_TIMEOUT	Disconnected	Occurs when idle time without executing any query while connected to DB exceeds this setting
UTRANS_TIMEOUT	Disconnected	Occurs when this setting value is exceeded until commit/rollback is executed after executing the query while connected to the DB (execution query is rolled back)

If the user has a 5-second timeout for a connection attempt and a blocking phenomenon on the network occurs, and there is no response within 30 seconds if the user wants to disconnect the connection, add the option to the connection string as follows.

```
DSN=127.0.0.1;CONNTYPE=2;TIMEOUT=5;CONNECTION_TIMEOUT=30
```

Other TIMEOUT settings operate in the unit set in $ALTIBASE_HOME/conf/altibase.properties, and can be controlled by executing the following query for each session. (When it is set to "0", it operates as infinite)

```
ALTER SESSION SET QUERY_TIMEOUT = 30 (Unit: second)
ALTER SESSION SET FETCH_TIMEOUT = 30
```

If the error by TIMEOUT occurs, the following error can be checked in the application.

Timeout Type	Error Message
CONNECTION_TIMEOUT	Connection time out
TIMEOUT	Client unable to establish connection
QUERY_TIMEOUT	Client's query exceeded in the execution time limitation
FETCH_TIMEOUT	The session has been closed by the server

© Altibase Corp.

Timeout Type	Error Message
IDLE_TIMEOUT	The session has been closed by the server
UTRANS_TIMEOUT	The session has been closed by the server

TIMEOUT-related errors are also recorded in $ALTIBASE_HOME/trc/altibase_boot.log.

```
[2010/07/06 13:10:35] [Thread-182894171744] [Level-1]
[Notify : UTrans Timeout] Session Closed by Server : Session ID = 53
CLIENT_INFO => TCP 127.0.0.1:3992(PID : 13645)
Time Limit => 3
Running Time => 5
Last Query => insert into t1 select * from t1 limit 1
Caused by Transaction => 82368
```

If the above log is recorded in altibase_boot.log, the developer must check and take action to determine why the Timeout error occurred.

Each cause is the same as described above, so if it occurs as a bottleneck of query processing, the corresponding query should be tuned. If UTrans_Timeout occurs as in the above example, commit/rollback is not performed after a change transaction occurs. The transaction control part of the program must be corrected.

If it is difficult for the user to solve it with the above instruction and need to force the processing, the TIMEOUT setting value for the DB session of the program should be processed rather than the property value of the entire DB.

Considerations when using SELECT Cursor

When implementing a SELECT statement using CURSOR, the user must perform an error check on the FETCH statement and close the CURSOR after an error occurs or after reading all data. If the user tries to FETCH a CURSOR that has already been read or try to work on a CURSOR that is not closed yet, an unintended error may occur. After using CURSOR, you must CLOSE normally.

Another consideration is that in the case of ALTIBASE, if COMMIT/ROLLBACK occurs during FETCH, the open CURSOR is automatically closed. Therefore, in order to create another transaction while FETCH is in progress, a separate DB session needs to be created and a separate transaction needs to be processed with the session or perform a COMMIT/ROLLBACK after all the FETCH is completed.

However, if there are a large number of records to be processed, it is recommended to implement a structure in which separate sessions are established and processed rather than bulk changes. If the number of records to be changed must be bound in one transaction, it is recommended to develop a program so that conditions can be separated and processed.

> **Info**
>
> Refer to the "Cautions during the bulk change operation" section below.

Checking SQL Error

It is recommended that developers of the ALTIBASE environment to check the SQL error after all queries are processed. In particular, the following case must be checked in order to find errors.

```
EXEC SQL PREPARE ….
EXEC SQL EXECUTE ….
If (SQLCODE != SQL_SUCCESS)
   Error_log ();

Or,
EXEC SQL DECLARE CURSOR…
EXEC SQL OPEN ….
If (SQLCODE != SQL_SUCCESS)
   Error_log ();
```

In the above code, there is no error checking after performing PREPARE or DECLARE CURSOR. In this case, if the PREPARE (DECLARE CURSOR) step causes an error, the SQL error in the EXECUTE (OPEN) step would cause an error such as "Not defined (XX)", and despite the fact that it was actually caused by the PREPARE (DECLARE CURSOR) error, but the error cannot be accurately confirmed.

ALTIBASE also sends a query to the server for the execution of PREPARE (DECLARE CURSOR), so it performs query processing up to the step before executing it.

It is important to note that SQL error checking must be performed when all SQL processing statements are entered.

Considerations for LOB type

When querying for LOB type in ALTIBASE, it must be operated in non-auto-commit mode. The following error may occur in auto-commit mode.

```
[ERR-91101 : Connection is in autocommit mode. One can not operate on LOB datas with autocommit mode on.]
```

It does not matter if the query is executed with the LOB in the column of the table, but the above error occurs when the LOB type is included in the SELECT target clause. Therefore, when querying the LOB type, it must be executed after changing the properties of the session to non-auto-commit mode.

However, in the case of using Altibase 6.3.1 or later JDBC driver, LOB type data can be queried in auto-commit mode by setting the transaction commit mode to be controlled on the client-side. How to set the transaction commit mode on the client-side of the JDBC is as follows.

- Example of altibase_cli.ini

```
[ DataSource ]
Server=192.168.1.1
Port=20300
User=SYS
Password=MANAGER
AlternateServers=(192.168.1.2:20300,192.168.1.3:20300)
ConnectionRetryCount=3
ConnectionRetryDelay=5
LoadBalance = On
```

© Altibase Corp.

```
SessionFailOver = Off
ClientSide_Auto_Commit = On
```

Using Prepared Statement

ALTIBASE stores the execution plan of internally executed queries to speed up the performance of query processing. However, if a query of PREPARE/EXECUTE is executed every time, the execution plan that has already been PREPARE is used internally, but because the network cost to execute PREPARE is queried every time, it is recommended to write the following code to prevent the downgrade of performance.

(O)	(X)

Prepare statement
while (1)
{
Execute statement
}
	while (1)
	{
	Prepare statement
	Execute statement
	}

There is a DB development library called OTL among open sources (Oracle, odbc template library). This library processes all queries in the form of prepare/execute every time. However, once the streaming option is activated, once prepared queries are only processed in the form of executing. The developer can query with the use of these tuning elements (reducing the network transmission cost). In the process, the dynamic should be minimized as much as possible and make sure that it is not preparing the same query every time.

```
sprintf (sql,  “INSERT INTO t1 VALUES (%d, %d)”, szNum1, szNum2);
EXEC SQL Prepare S for :sql;
EXECUTE S;
```

The above code creates an SQL statement to be executed each time, prepares it, and executes it. In this case, the following would be reasonable.

```
EXEC SQL INSERT INTO t1 VALUES (:szNum1, :szNum2);
```

SQL Tuning

When developing operations related to DB operation, there are cases where query processing is slow. Or, there are cases where the query was processed quickly in the test environment, but the query process is very slow in the integration test or operation stage. This can be seen as a case where the developer or DBA neglects the verification or testing process of the execution plan for all queries executed.

ALTIBASE can check the query processing process, that is, the execution plan as follows.

```
ALTER SESSION SET EXPLAIN PLAN = ON;
```

© Altibase Corp.

After executing the above command, the execution plan can be checked by executing a query in iSQL.

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
Alter success.
iSQL> SELECT a, COUNT(*) FROM t1 WHERE a = 1 GROUP BY a;
A COUNT
-----------------------------------
1 262145
1 row selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 2, TUPLE_SIZE: 16 )
GROUP-AGGREGATION ( ITEM_SIZE: 32, GROUP_COUNT: 1, BUCKET_COUNT: 1024, ACCESS: 1, SELF_ID: 3, REF_ID: 2 )
SCAN ( TABLE: T1, FULL SCAN, ACCESS: 262145, SELF_ID: 2 )
------------------------------------------------------------
```

We can see that the above execution plan FULL SCAN table T1 and accessed 262,145 records. If the index exists and the condition to use the index is specified, the following might be displayed.

```
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 1, TUPLE_SIZE: 4 )
SCAN ( TABLE: T1, INDEX: IDX_T1, ACCESS: 2, SELF_ID: 2 )
------------------------------------------------------------
```

For more information on SQL tuning, please refer to the "Performance Tuning Guide" manual.

Cautions during the bulk change operation

Since general bulk change operation has a high load not only in ALTIBASE but also in other DBMS, there are precautions required for each vendor. ALTIBASE requires the following cautions.

① Possibility of memory increase in case of mass change of memory table

ALTIBASE supports the MVCC (Multi-Version Concurrency Control) technique. In the MVCC technique, when a change occurs, the same record is duplicated in the empty space of the table to make the change. If 10 million changes are made, memory increases due to 10 million copies will inevitably occur. (Of course, the increased area is reused as data or replication area, but the memory is not released in real time.)

② Continuous increase of transaction log files by performing bulk change operation

A large number of changes require a redo log for each change target record to be recorded in the transaction log file. Therefore, it is necessary to record the redo log of all the records that are changed in one transaction, but if the scope of this target is large, as many transaction log files are created as well. (ALTIBASE deletes the transaction log file at the time of the checkpoint. If there is information on which the transaction is ongoing in the log file, it cannot be deleted, and there is a possibility of a disk shortage failure due to the log file increase.)

③ Possibility of dead-lock occurrence and delay in the reflection in a replication environment

In a replication environment, as described above, the change information of all records changed by one transaction is transmitted to the other server.

© Altibase Corp.

In this process, a lock is held for the corresponding record of the other server. If the other server changes the record in the same range, a deadlock may occur between the transaction caused by the redundancy log and the transaction occurring in the other server itself.

© Altibase Corp.

04

Altibase Trace Logs

ALTIBASE records trace logs for important status information and errors for each module in the $ALTIBASE_HOME/trc path, which is the installed path.

Classification of trace logs

The trace log recorded by ALTIBASE is as follows.

Classification	Description
altibase_boot.log	Logs related to start/stop processes and overall DBMS operations
altibase_sm.log	Logs related to checkpoints and tablespaces
altibase_qp.log	Logs related to DDL execution and query processing
altibase_rp.log	Logs related to replication operation status
altibase_rp_conflict.log	Logs related to replication conflicts
altibase_dk.log	Logs related to DB link usage
altibase_xa.log	Logs related to XA transactions
altibase_mm.log	Logs generated during main module processing
altibase_error.log	Logs related to errors occurring in the server
altibase_dump.log	Debugging logs for errors occurring in the server
altibase_cm.log	Logs warning or trace messages from the communication module
altibase_lb.log	Logs warning or trace messages from the load balancer
altibase_job.log	Logs information related to the execution of JOB objects
altibase_ipc.log	Logs information about resources created during IPC connection
altibase_ipcda.log	Logs information about resources created during IPCDA connection
altibase_snmp.log	Logs warning or trace messages from SNMP
killCheckServer.log	Logs the results of executing the killCheckServer utility

By default, the log files are circulated to altibase_xx.log-[1-10], including altibase_xx.log, to record 10 log files.

In the following, messages that are meaningful to the user are explained separately for each trace log.

altibase_boot.log

The user must check altibase_boot.log first when an error occurs in DBMS. The **orange color** mark in the table is an error that the user must resolve.

© Altibase Corp.

Log Messages and Descriptions
ERR-0108d (errno=0) License is invalid or expired.
This error occurs when the license file is missing or incorrect when starting up. The user must check the license information in ALTIBASE and have it reissued if necessary.
ERR-01052(errno=13) Unable to invoke open() function on [/dbs/mydb-0-0]
This error occurs when the DB has not been created before the initial operation or when there is no permission to access the file. The user must create a DB or check if the user has permission to access directories and files. Alternatively, it may be the case that the data file was deleted by mistake by the user. In this case, a procedure to recover using a backup copy is required.
ERR-01052(errno=2) Unable to invoke open() function on [/dblog01/logfile536358]
During the Altibase operation phase, automatic recovery is performed, which occurs when the redo log file required for this process is damaged or does not exist. The user must request technical support from the ALTIBASE HQ.
ERR-0001c(errno=76) Unable to shutdown the communication channel
During the ALTIBASE shutdown phase, a warning message from the session-related cleanup process has no effect on the operation.
ERR-71019 (errno=104) Failed to invoke a system function, write() (read, accept, select)
This error occurs by an error in a disk or network. The user must check the file system or make sure there are no network problems. Alternatively, system calls may fail due to insufficient memory.
ERR-40029 (errno=16) Failed to invoke a system function, flock_trywrlock()
This error occurs when ALTIBASE is already running or ALTIBASE is restarting while it is already running. Or, it can occur when the system call fails due to insufficient memory.
ERR-11018 (errno=2) The version of data file for backup is not compatible with the version of storage manager. Backup DB => [Version ID = 5.4.1, Bit = 64, Endian = LITTLE LogSize = 10485760 Transaction Table Size = 1024] Server=>[Version ID = 5.4.1, Bit = 64, Endian = LITTLE LogSize = 10485760 Transaction Table Size = 2049]
This error occurs when attempting to recover with a data file that is incompatible with the ALTIBASE version currently in use while running with the backed-up data. Or, this occurs when attempting to run in a different state from the ALTIBASE setting value set at the time of backup.
[ERROR] first write () operation failed. errno=32
This error occurs in the read/write system call while the session is disconnected, so the information is recorded in the form of a warning. This log is also recorded when an application forces a shutdown without a normal disconnection command. The operation is not affected.
ERR-7101d(errno=11) Protocol header error. (TCP 211.115.75.212:63847) Protocol processing failed. Close connection...
When a session accessed with the ALTIBASE connection port attempts to connect with a version that is not compatible with the ALTIBASE server version, the user must traceback with the output connection information to check whether the server/client versions are compatible. The operation is not affected.
ERR-4000f(errno=2) No Error Message Loaded
This error occurs when the files located in the $ALTIBASE_HOME/msg/ path do not exist or are not compatible with the version. Or, this occurs when the user does not have permission to read on the path.

© Altibase Corp.

Log Messages and Descriptions
[Notify : Query Timeout] Query Canceled by Server : Session ID = 102373 **[Notify : Fetch Timeout] Session Closed by Server : Session ID = 80106** **[Notify : Utrans Timeout] Session Closed by Server : Session ID = 80106** **[Notify : Idle Timeout] Session Closed by Server : Session ID = 80106**
When TIMEOUT occurs due to QUERY/FETCH/UTRANS/IDLE TIMEOUT setting, a warning type log is recorded. If there is a problem, the session information is recorded along with the message, so the user must track the session and resolve it.
ERR-01027(errno=0) No more IPC channel (MAX=50, USED=50, BUFSIZE=65536)
This error occurs when attempting more IPC access than available resources.
[Warning] Memory allocation failed. Size:524320 Timeout: 0 **ERR-61055(errno=12) Memory allocation failure**
This error occurs when an internally required memory allocation fails due to insufficient memory. In this case, the user must determine the cause of insufficient memory and resolve it. If a case of insufficient swap memory occurs, ALTIBASE may terminate abnormally without warning.
ioctl() failed
This error occurs when there is a temporary problem with performing system calls between socket or disk I/O operations. Generally, this can be ignored, but it is recommended to check the network or disk.
WANING : write error number : 0, file name : /dblog01/logfile537572, file handle : 440, intended size : 10485760, writed size : 10485760 **또는** **File Extending Failed. : The disk space has been exhausted.** **Error : Extending DataFile : from 0, size 10485760**
When creating a file that ALTIBASE requires due to insufficient disk capacity, this is a warning that the file could not be created normally due to insufficient capacity. The user must check the disk capacity and resolve it.

altibase_sm.log

altibase_sm.log is an important basis for determining whether checkpoints are being executed and whether there are long-running queries.

Log Messages and Descriptions
[CHECKPOINT-BEGIN]
This means the checkpoint has started.
Remove Online Log File at LFG [0]: File[220 ~ 227]
This message removes the transaction log files that are no longer needed for recovery as the checkpoint is performed. If it is continuously displayed as File [None], it is necessary to check whether there are any queries that have been executed for a long time or that there are no untransmitted parts due to the replication.
[CHECKPOINT-summary] BeginChkptLSN=[0,83,9820280], EndChkptLSN=[0,83,9820721], DiskRecLSN=[0,83,9820280]
When the checkpoint has been successfully completed, summary information about checkpoint performance information is recorded in the log.

© Altibase Corp.

Log Messages and Descriptions
[CHECKPOINT-END]
This means the checkpoint has been successfully completed.
Minimum LSN = [0,83,9824136]
The above value is the offset position of the log file being recorded of the transaction in progress. Therefore, if that value does not change and keeps the same value, it is necessary to check whether there is a query that is executed for a long time as in the Remove log above, or whether the replication is not transmitted.
Database-Level Backup Completed [SUCCESS]
This means the backup has been successfully completed.
DISK TABLESPACE .... DATABASE ....
This means the backup of disk tablespaces has been performed.
~/loganchor0 BACKUP TO
This means that the backup of the log anchor has been performed.
MEMORY TABLESPACE DATAFILE .... BACKUP TO ....
This means that the backup of memory tablespace has been performed.

altibase_qp.log

In altibase_qp.log, a log of the DDL executed by the user and the case of changing DB property values with “ALTER SYSTEM” statement, etc. are recorded. They are classified as follows.

Log Messages and Descriptions
[EXEC_DDL_BEGIN : create index T1_IDX on T1 (A)]
Log indicating the start of DDL executed by the user
[EXEC_DDL_END : SUCCESS]
Log indicating the normal completion of DDL executed by the user
[EXEC_DDL_END : FAILURE] errorcode 1627549735.
Log indicating the error details of DDL executed by the user
[SET-PROP] CHECKPOINT_BULK_WRITE_PAGE_COUNT=[0]
Log indicating when the user changes the DB property setting value that affects the entire system

altibase_rp.log

Because altibase_rp.log records the operation state of replication and queries when various conflicts occur, it provides important tracks for replication problems.

Log Messages and Descriptions
ERR-31017(errno=0) Replication not found

© Altibase Corp.

Log Messages and Descriptions
This error occurs when the replication target objects do not exist when executing DDL. Therefore, it is necessary to check whether the replication target objects exist.
ERR-6200f(errno=16) [Sender] Stop sender thread REP1 at [9765213], Restart SN[9765207]
This log is created when the replication sender is explicitly stopped by the user
ERR-71017(errno=79) Failed to invoke a system function, connect() **ERR-61012(errno=79) [Sender] Failed to connect to the peer server** **ERR-61022(errno=79) [Sender] Sender Sleep : 60 seconds** **[Sender] getNextLastUsedHostNo: from 192.168.1.81:53341 to 192.168.1.81:53341** **ERR-61003(errno=9) Unable to read from a socket** **ERR-61012(errno=119) [Sender] Failed to connect to the peer server** **ERR-6100d(errno=119) [Sender] Failed to handshake with the peer server (Handshake Process Error)**
This is the log when the replication sender cannot connect to the other side's server and attempts to reconnect. It is necessary to check whether the other side's server, network status, and ALTIBASE are running properly. If the other party's replication is explicitly dropped, it is a normal error.
[Recovery Sender] Replication REP1 Start... at [60495917]
This is the log when the replication and the sender have started.
[Receiver] Replication REP1 Started ...
This is the log when receiver and replication have started properly.
RECEIVER:REPLICATION STOP MSG arrived! **[ReceiverApply] REPLICATION STOP XLog arrived!** **Normal Stop!** **[Receiver] Replication REP1 Stopped ...**
This is when the replication of other server is explicitly stopped by the user.
ERR-61047(errno=0) [Receiver] REP1 receiver has error in recvXlog() **ERR-61048(errno=0) [Receiver] REP1 receiver has recvXLog error in run()** **ERR-6104b(errno=0) [Receiver] REP1 receiver is ended (by thr_exit)**
This error occurs when the replication of the other server is explicitly stopped by the user.

The name of the replication object used in the example above is “REP1”. Since the name of the replication object specified by the user will be different, it should be considered when searching for errors, and it is recommended to detect the operation state of the replication Sender/Receiver and conflict errors.

altibase_rp_conflict.log

The altibase_rp_conflict.log records SQL statements executed when various conflicts occur, providing important clues for troubleshooting replication issues.

Log Messages and Descriptions
ERR-61036(errno=0) [Receiver] err_not found in deleteXlog() **ERR-61000(errno=0) The received record is not found in the database.**

© Altibase Corp.

Log Messages and Descriptions
In the process of executing the DELETE transaction received through replication, if the corresponding record does not exist, this conflict error is recorded.
ERR-61035(errno=0) [Receiver] An update conflict encountered. **ERR-61001(errno=0) A conflict has been occurred while executing the received statement.**
This error occurs when a conflict error is recorded because the before/after values of the corresponding record are different during the process of executing the UPDATE transaction received through replication.
ERR-6103a(errno=0) [Receiver] err_not_found in updateXlog() **ERR-61000(errno=0) The received record is not found in the database.**
In the process of executing the UPDATE transaction received replication, if the corresponding record does not exist, this conflict error is recorded.
ERR-11058(errno=0) The row already exists in a unique index.
In the process of executing the INSERT received through replication, the corresponding record already exists on the receiving side. (There are data with the same PK)

In addition, if a conflict occurs, it is important to find the cause of the conflict with the SQL statement that remains when the conflict occurs, because there is a possibility that data inconsistency may have occurred due to either not properly complying with the considerations of the replication environment described above.

altibase_dk.log

In altibase_dk.log, various information and errors that may occur in the environment using the Database Link are recorded.

Log Messages and Descriptions
ERR-b1070(errno=0) ODBC problem occurred [0:IM002:[unixODBC][Driver Manager]Data source name not found, and no default driver specified]
This error occurs when the specified data source name is not found or LD_LIBRARY_PATH is not set.
ERR-b1070(errno=239) ODBC problem occurred [327730:08001:Client unable to establish connection.]
This error occurs when it is not possible to connect to the Altibase server or when the odbc.ini environment variable is not set. This can be resolved by checking the server running state and adding odbc.ini to the environment variable.

altibase_error.log

The altibase_error.log provides tracks for analyzing the cause of abnormal termination of ALTIBASE.

© Altibase Corp.

Log Messages and Descriptions
BEGIN-STACK-[CRASH]===================================== Caller[0] 0000000000417208 Caller[1] 00007FFB5324FC20 Caller[2] 000000000042D81D Caller[3] 0000000000E72EEB Caller[4] 000000000042944A Caller[5] 000000000042AE2F Caller[6] 000000000042B7BC Caller[7] 0000000000EA65FB Caller[8] 00007FFB5324517A Caller[9] 00007FFB52214DC3 END-STACK===================================== [========= FATAL Terminated ==========]

When ALTIBASE is abnormally terminated due to a bug, its own trace log is outputted in which process occurred. If this error occurs, the ALTIBASE process is terminated. First, backup all files in $ALTIBASE_HOME/trc/. Then backup the redo log files and, if there is enough disk space, even get the database file. When such file backup is completed, resolve it by restarting ALTIBASE and contact ALTIBASE for technical support.

© Altibase Corp.

05

CLIENT APPLICATION Error Messages

This chapter describes the types of errors that occur frequently in the environment in which an application program using ALTIBASE, and also explains what users have to check for each type.

Connection does not exist

This error occurs when attempting to process a query in the state that the DBMS is not connected or disconnected. This can occur in the following cases:

1. When executing a query after disconnection by the TIMEOUT policy described previously
2. In the case of a thread program, a connection unique name that does not exist is used

Check the above details and take appropriate action for the related matters. For example, if it is caused by the TIMEOUT policy, it is necessary to take measures such as increasing the TIMEOUT setting value of the session or performing query tuning.

Communication link failure

Generally, the above "Connection does not exist" occurs after this error. This error means that the session was disconnected while executing the query. The cause of this is generally released by the TIMEOUT policy as described previously. Occasionally, the connection may be disconnected due to a network problem. Because this problem is a communication socket problem and can be caused by various reasons, it is recommended to check the following:

1. Check whether there is a log in altibase_boot.log where the session was disconnected.
2. In the case of a threaded program, check whether the precautions to be followed are not properly observed.
3. Check if there was any temporary problem on the network.

Calculate stack overflow

This error occurs when the stack required by the query executed as an object used in ALTIBASE internally processing a query is insufficient. The user can process the query normally by executing the query as follows before executing the query and then executing the query with an error.

```
ALTER SESSION SET STACK SIZE = 8192;
```

Because this value causes an increase in memory, it is recommended to change and use only the relevant session rather than reflecting it on the entire system.

© Altibase Corp.

Conversion not applicable

This error occurs in the following cases.

```
CREATE TABLE t2 (a BYTE (2));
INSERT INTO t2 VALUES  (CLOB'1');
```

As in the example above, when CLOB data is put in the BYTE column, an error occurs because the type conversion cannot be converted from CLOB to BYTE. That is, there is a matrix compatible with each column type, but this error occurs when such incompatible queries are executed.

For compatibility between column types, please refer to the "General Reference" Manual.

Invalid cursor state

This occurs when attempting to open CURSOR again without normally closing CURSOR. Or, it occurs when attempting to fetch a CURSOR that has already been fetched again.

Occasionally, if the user program is a thread, it may not properly control concurrency for the connection object and thus handle a cursor that is not valid in its own thread. The user must resolve the problem by using it.

```
DECLARE CURSOR
OPEN CURSOR
FETCH CURSOR
/ CLOSE CURSOR / -- Omission cases
```

Not defined cursor

This error occurs when using CURSOR and the CURSOR NAME is specified, but this CURSOR NAME does not exist. Or, this error occurs in the process of executing DECLARE (PREPARE) CURSOR. The user must develop as follows.

```
DECLARE CURSOR1 FOR :statement;
If (SQLCODE != 0)              -- Error must be checked even in the PREPARE stage
    Error_process();
OPEN CURSOR1 USING :variable;
```

Invalid request to process the SQL statement

Generally, this error occurs frequently in thread-structured programs, and also occurs when the order of protocols exchanged between the ALTIBASE server and the client-server is different when the concurrency control for the thread-structured connection object is incorrect as described above.

If it is in the thread-structure program, the user needs to make sure that this concurrency control is correct. Alternatively, the user should check whether the order in which queries should be executed, such as PREPARE->BINDING->EXECUTE, is not properly followed.

© Altibase Corp.

Invalid literal

This error occurs when an error such as entering a character into a numeric type.

```
CREATE TABLE t1 (a INTEGER);
INSERT INTO t1 VALUES (‘AAAA’);
```

This occurs when an attempt is made to insert data as a string even though the column type is numeric, as in the example above.

Conversely, if the user tries to insert a string into a numeric variable, an error occurs as follows.

```
Invalid character value for cast specification.
```

Invalid length of data type

This error occurs when data is input with a larger value than the column type.

```
CREATE TABLE t1 (a CHAR(2));
INSERT INTO t1 VALUES ('aaaa');
[ERR-2100D : Invalid length of the data type]
```

As in the example above, the column type can store up to 2 bytes. If the user tries to input 4 bytes, an error occurs because the data value is larger than the specified column size.

Indicator variable required but not supplied error

This error occurs in the development environment using the precompiler. If the return value of the column is returned as NULL in the SELECT result, this error message occurs along with SQL_SUCCESS_WITH_INFO if the INDICATOR variable is not specified.

```
int IND_C1;
int IND_C2;
EXEC SQL SELECT C1, C2
INTO :H_C1 INDICATOR :IND_C1 , :H_C2 INDICATOR :IND_C2;
```

As in the above case, you can explicitly use the INDICATOR variable or use the -unsafe_null option in the Precompiler option to prevent an error from occurring.

Invalid size of data to bind to host variable Data Size …

This error occurs when a query is processed by inserting data larger than the size of the host variable. For example, it occurs when a string larger than 20 is inserted into a host variable declared as CHAR (20). Generally, it can occur when the user fails to initialize a variable or garbage value comes in. Alternatively, it may occur when the binding of statement B is executed after the PREPARE of statement A because of a concurrency control failure in a thread environment program. (Statement A needs to bind 20 bytes, but is handled incorrectly and binds 100 bytes of Statement B). First, the user must output the host variable

© Altibase Corp.

value to check that it does not exceed the length, and check that there is no problem in the concurrency part in the handling of the thread part.

Too many pages are allocated

For ALTIBASE memory tablespaces, the sum of all memory tablespaces cannot exceed MEM_MAX_DB_SIZE in $ALTIBASE_HOME/conf/altibase.properties. This error message occurs because all space is full. Therefore, it is necessary to delete unnecessary tables or increase MEM_MAX_DB_SIZE to a larger value and restart ALTIBASE. After these temporary measures, it is necessary to resolve it by checking for some reason the increase in the usage of the memory tablespace and whether there has been a query or mass change of the usage for each table.

The Tablespace does not have enough free space

This error occurs when there is insufficient free space in the tablespace. In the case of a disk tablespace, it is necessary to add data files.

The transaction exceeds lock timeout specified by user

DML waits for a lock and operates according to the TIMEOUT policy set in the server, but this error occurs immediately when DDL waits due to a lock held by DML.

Time	Session A	Session B
T1	INSERT INTO x1 VALUES … (Lock획득)	
T2		ALTER TABLE x1 ADD COLUMN …(Error)

As shown in the table above, if session B attempts to execute DDL while session A acquires the lock with DML first, this error occurs. Therefore, check if a lock exists in the table and resolve it before proceeding.

The update log size '…' is bigger than TRX_UPDATE_MAX_LOGSIZE '…'

In ALTIBASE, this error occurs when the amount of redo log created by a change transaction is greater than the value specified in TRX_UPDATE_MAX_LOGSIZE in $ALTIBASE_HOME/trc/altibase.properties to prevent subsequent failures caused by the bulk change operation. If the transaction must be executed as a single query, the user must change this property value at the session-level and process it as follows.

Default value: 10485760 (bytes)

```
ALTER SESSION SET TRX_UPDATE_MAX_LOGSIZE = 0;  (Settings that remove the restriction)

ALTER SESSION SET TRX_UPDATE_MAX_LOGSIZE = 52428800;  (Set the limit to 50 MB.)
```

Please be cautious when processing such changes, as more redo log files are created and a situation such as a disk full failure may occur. Additionally, ALTIBASE attempts to acquire the X-Lock on the table to prevent the increase of resources due to MVCC when making bulk changes, so users need to be aware that queries and changes to the table will be queued due to the X-Lock.

© Altibase Corp.

For more detailed information, refer to "Error Message ERR-11118"

String data right truncated

This error occurs when the value of the character type returned from the SELECT statement is larger than the size of the declared host variable. Check the column size of the table to be accessed by accessing the DB, and declare the size of the used host variable as (column size + 1 byte).

Value overflow

This error occurs when a value larger than the expression range of the numeric column is input.

```
CREATE TABLE x2 (a NUMERIC(10, 5));
insert into x2 values (123444.5678);
[ERR-21010 : Value overflow]
```

Several statement still opened

This error occurs when a query such as CREATE TABLE is executed while the fetch protocol is in progress. Therefore, DDL must be executed after CURSOR-FETCH is completed in all the same sessions, or DDL queries must be executed after closing the open CURSOR. Alternatively, it is necessary to change the code so that the DDL query is executed in a separate session.

© Altibase Corp.