

# ODBC User's Manual

---

## Altibase 7.3

Altibase® Application Development

Altibase Application Development ODBC User's Manual

Release 7.3

Copyright © 2001~2023 Altibase Corp. All Rights Reserved.

This manual contains proprietary information of Altibase® Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright patent and other intellectual property law. Reverse engineering of the software is prohibited.

All trademarks, registered or otherwise, are the property of their respective owners.

**Altibase Corp**

10F, Daerung PostTower II,

306, Digital-ro, Guro-gu, Seoul 08378, Korea

Telephone : +82-2-2082-1000

Fax : +82-2-2082-1099

Customer Service Portal : <http://support.altibase.com/en/>

Homepage : <http://www.altibase.com>



# Table Of Contents

---

- [Preface](#)
  - [About This Manual](#)
- [1. Introduction](#)
  - [Introduction](#)
  - [Data Types](#)
  - [ODBC API](#)
- [2. Installing and configuring the Altibase ODBC Driver](#)
  - [Installing the Altibase ODBC Driver](#)
  - [Configuration](#)
- [3. ODBC Programming](#)
  - [Connection String](#)
  - [Basic Programming Example](#)
  - [Example of Using LOB](#)

# Preface

---

## About This Manual

This manual describes how to install and configure the Altibase ODBC Driver.

## Types of Users

This manual has been prepared for the following Altibase users:

- Database managers
- Performance managers
- Database users
- Application developers
- Technical support engineers

It is recommended for those reading this manual possess the following background knowledge:

- Basic knowledge in the use of computers, operating systems, and operating system utilities
- Experience in using relational databases and an understanding of database concepts.
- Computer programming experience
- Experience in a database server management, operating system management, or network administration

## Organization

This manual is organized as follows:

- Chapter 1: Introduction
- Chapter 2: Installing and Configuring the Altibase ODBC Driver
- Chapter 3: ODBC Programming
- A. Appendix: FAQ

## Altibase Welcomes Your Comments.

Please let us know what you like or dislike about our manuals. To help us with future versions of our manuals, please tell us about any corrections or classifications that you would find useful. Please send it to our customer service portal (<http://support.altibase.com/en/>) with the following comments or feedback.

Include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

When you need immediate assistance regarding technical issues, please contact Altibase's Customer Support site (<http://support.altibase.com/en/>).

Thank you. We appreciate your feedback and suggestions.

# 1. Introduction

---

This chapter introduces the Altibase ODBC driver.

## Introduction

ODBC(Open Database Connectivity) is a standard programming language interface accessing databases, which is developed by Microsoft. In order that ODBC Application developed using ODBC API accesses a database, a specific module or driver(e.g., ODBC Driver Manager) the database providing is required. Altibase provides the Altibase ODBC driver, which conforms to ODBC 3.51 specifications, for this purpose.

As the Altibase ODBC driver is built upon Altibase CLI, you can refer to the Altibase CLI User's Manual for more detailed information on the internal procedures of the Altibase ODBC driver or for more specific description.

## Data Type

For more detailed information on on the mapping relationship between SQL data type and ODBC data type supported by Altibase, please refer to "B. Appendix: Data Types" in Altibase CLI User's Manual.

## ODBC API

For more detailed further information on the ODBC API, please refer to the ODBC API Reference (<http://msdn.microsoft.com/en-us/library/windows/desktop/ms714562%28v=vs.85%29.aspx>).

## ODBC Conformance Level

Based on the conformance of ODBC functions, this section describes three types of functions: those that are currently supported for Altibase ODBC, those that are soon to be supported, and those that are not supported.

The purpose of specify the ODBC conformance level is to provide to the application, in- formation of which functionality of the ODBC driver to use. ODBC conformance is currently categorized into the three levels: Core, Level 1 and Level 2. To satisfy the conformance level of a function, the driver must meet all the conditions required for that level.

The following table shows the conformance level of the ODBC 3.x standard. This is different from the confomance level of ODBC 2.x; Level 1 for ODBC 2.x can be seen here as a core.

The Altibase ODBC driver conforms to ODBC 3.51 specifications. The following table shows the conformance level of each ODBC function, and whether the Altibase ODBC driver is supported or not.

Function Name	Levels	Support Status	Future Support	Remarks
SQLAllocHandle	Core	O		
SQLBindCol	Core	O		

Function Name	Levels	Support Status	Future Support	Remarks
SQLBindParameter	Core	O		
SQLBrowseConnect	Level1	X	X	
SQLBulkOperations	Level1	O		
SQLCancel	Core	O		
SQLCloseCursor	Core	O		
SQLColAttribute	Core	O		
SQLColumnPrivileges	Level2	X	X	Columns are not supported by Altibase
SQLColumns	Core	O		
SQLConnect	Core	O		
SQLCopyDesc	Core	X	O	
SQLDescribeCol	Core	O		
SQLDescribeParam	Level2	O		Not fully supported.
SQLDisconnect	Core	O		
SQLDriverConnect	Core	O		
SQLEndTran	Core	O		
SQLExecDirect	Core	O		
SQLExecute	Core	O		
SQLFetch	Core	O		
SQLFetchScroll	Core	O		
SQLForeignKeys	Level2	O		
SQLFreeHandle	Core	O		
SQLFreeStmt	Core	O		
SQLGetConnectAttr	Core	O		
SQLGetCursorName	Core	O		
SQLGetData	Core	O		
SQLGetDescField	Core	O		ODBC 3.0



Function Name	Levels	Support Status	Future Support	Remarks
SQLGetDescRec	Core	O		ODBC 3.0
SQLGetDiagField	Core	O		ODBC 3.0
SQLGetDiagRec	Core	O		ODBC 3.0
SQLGetEnvAttr	Core	O		
SQLGetFunctions	Core	O		
SQLGetInfo	Core	O		
SQLGetStmtAttr	Core	O		
SQLGetTypeInfo	Core	O		
SQLMoreResults	Level1	O		
SQLNativeSql	Core	O		
SQLNumParams	Core	O		
SQLNumResultCols	Core	O		
SQLParamData	Core	O		
SQLPrepare	Core	O		
SQLPrimaryKeys	Level1	O		
SQLProcedureColumns	Level1	O		
SQLProcedures	Level1	O		
SQLPutData	Core	O		
SQLRowCount	Core	O		
SQLSetConnectAttr	Core	O		
SQLSetCursorName	Core	O		
SQLSetDescField	Core	O		ODBC 3.0
SQLSetDescRec	Core	O		ODBC 3.0
SQLSetEnvAttr	Core	O		
SQLSetPos	Level1	O		
SQLSetStmtAttr	Core	O		
SQLSpecialColumns	Core	O		

Function Name	Levels	Support Status	Future Support	Remarks
SQLStatistics	Core	O		
SQLTablePrivileges	Level2	O		
SQLTables	Core	O		

## 2. Installing and Configuring the Altibase ODBC Driver

---

This chapter offers explanations on how to install and configure the Altibase ODBC driver.

### Installing the Altibase ODBC Driver

This section describes how to install the Altibase ODBC driver.

#### Unix-like Operating Systems

For Unix-like operating systems, the Altibase ODBC driver is also installed when you install the Altibase server or client package. For more detailed information on how to install the Altibase server or client package, please refer to the Installation Guide.

If you install the 64-bit package, the following 32-bit and 64-bit ODBC drivers are installed to \$ALTIBASE\_HOME/lib.

```
libaltibase\_odbc-64bit-ul32.so: SQLLEN size is 32 bits
```

```
libaltibase\_odbc-64bit-ul64.so: SQLLEN size is 64 bits
```

The reason why two drivers are included in the 64-bit package is as follows. The size of the SQLLEN type is defined by 64-bit ODBC Driver Managers as 64 bits. However, as unixODBC (one of the ODBC Driver Managers available for use in Unix-like operating systems) defines the size of the SQLLEN type to 32 or 64 bits, depending on the version and compile option, Altibase provides two drivers to offer a wide range of support. When using unixODBC, it is recommended that you select the driver that matches the SQLLEN size.

When installing the 32-bit package, the following file is installed to \$ALTIBASE\_HOME/lib.

```
libaltibase_odbc.so
```

For HP operating systems, the file name is the same as above and has only an extension of sl.

### Configuration

This section describes how to configure the ODBC driver.

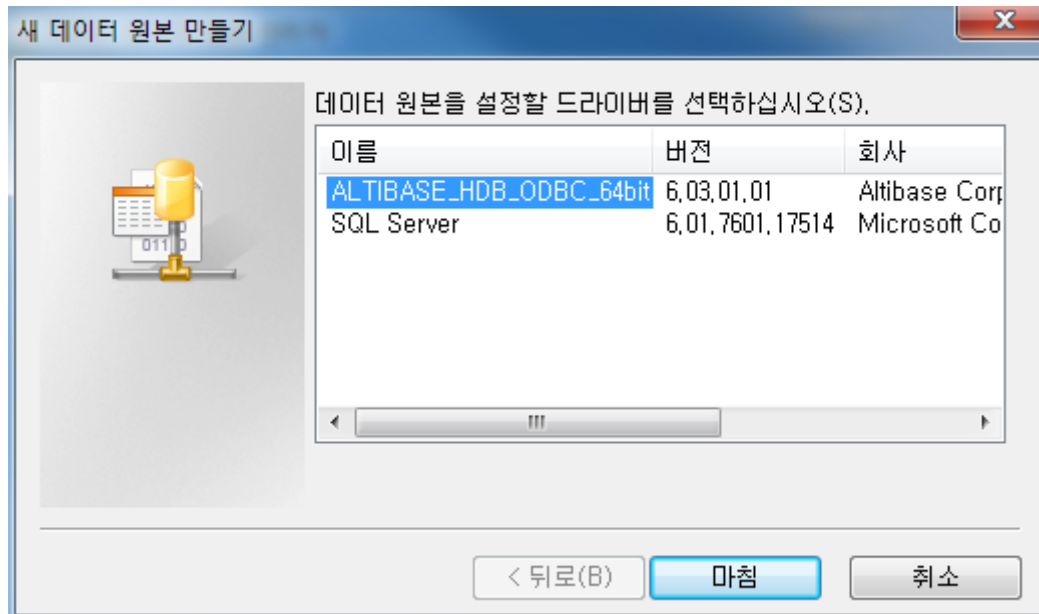
#### Unix-like Operating Systems

In order to use the ODBC driver in Unix, you must first install the ODBC Driver Manager. The unixODBC Driver Manager and iODBC Driver Manager are ODBC driver managers available for use in Unix. For more detailed information on each driver manager, please refer to the following links.

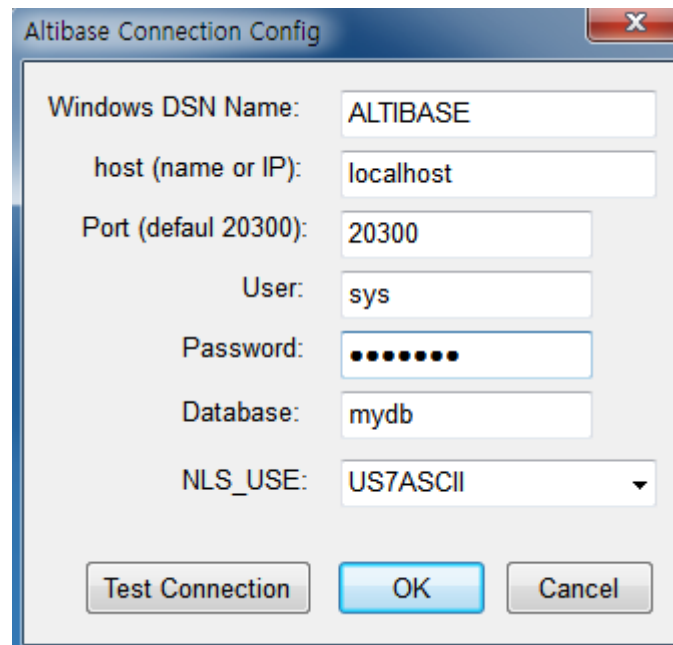
- <http://www.unixodbc.org/>
- <http://www.iodbc.org/>

## Adding DSN

For an ODBC application to obtain access to the database, the DSN of the database must be added. The following dialog box appears when you click “Add” on the [Data Source(ODBC)] User DSN or System DSN tab. Then, select ODBC driver and click “Finish”.



When the “Altibase Connection Config” window appears, enter the following.



- Windows DSN Name: Enter the data source name
- Host (name or IP): Enter the host name or IP address of the computer on which the Altibase server to be accessed is running.
- Port (default 20300): Enter the listening port number of the Altibase server. You can check the PORT\_NO value from the altibase.properties file or the value of the ALTIBASE\_PORT\_NO environment variable.
- User: Enter the database user name.
- Password: Enter the database user password.

- Database: Enter the database name.
- NLS\_USE: Enter the client character set.

You can check whether or not the ODBC driver is successfully connected to the database by clicking “Test Connection”. Once you click “OK”, you will be able to see the data source added to the DSN tab in the name you have just entered.

## 3. ODBC Programming

This chapter describes how to develop ODBC application using the Altibase ODBC Driver with examples.

### Connection String

When developing an ODBC application using Altibase ODBC, you can also use a connection string, instead of a DSN. The connection string consists of the following attributes:

Attributes	Description
DRIVER	ODBC driver name. This can be checked in the ODBC data source administrator window
User	The database user name
Password	The database user password
Server	The IP address of the Altibase server to be connected
PORT	The listening port number of the Altibase server
NLS_USE	The client character set
LongDataCompat	ON or OFF It is recommended to set it as ON when using large data such as BLOB. The default is OFF.

The following is an example of a connection string constructed using the above properties:

```
"DRIVER=ALTIBASE_HDB_ODBC_64bit;User=SYS;Password=xxx;Server=127.0.0.1;PORT=20300;NLS_USE=US7ASCII;LongDataCompat=OFF"
```

### Basic Programming Examples

Below are example codes of an ODBC application connecting to the Altibase server and its execution results.

#### Example

```
/* test_odbc.cpp */
#include <windows.h>
#include <sql.h>
#include <sqlext.h>
#include <stdio.h>
#include <stdlib.h>

#define SQL_LEN 1000
#define MSG_LEN 1024
```

```

SQLHENV      henv;
SQLHDBC      hdbc;
SQLHSTMT     hstmt;
SQLRETURN    retcode;

void execute_err(SQLHSTMT stat, char* q)
{
    printf("Error : %s\n",q);
    SQLINTEGER errNo;
    SQLSMALLINT msgLength;
    SQLTCHAR errMsg[MSG_LEN];

    if (SQL_SUCCESS == SQLError ( henv, hdbc, stat, NULL, &errNo, errMsg, MSG_LEN,
&msgLength ))
    {
        printf(" Error : # %lld, %s\n", errNo, errMsg);
    }

    SQLFreeStmt(stat, SQL_DROP);
    if (SQL_ERROR == SQLDisconnect(hdbc))
    {
        printf("disconnect error\n");
    }

    SQLFreeConnect(hdbc);
    SQLFreeEnv(henv);

    exit (1);
}

void main()
{
    char      *DSN, *DBNAME, *USERNAME, *PASSWD, *PORTNO;
    char      query[SQL_LEN], name[21];
    int       age;

    SQLCHAR constr[100];
    SQLINTEGER len;
    DSN = "ALTIBASE"; // Domain Server Name

    /* Allocate memory for the Environment */
    if(SQLAllocEnv(&henv) == SQL_ERROR)
    {
        printf("AllocEnv error!!\n");
        exit(1);
    }

    /* Allocate memory for a connection */
    if(SQLAllocConnect(henv, &hdbc) == SQL_ERROR)
    {
        printf("AllocDbc error!!\n");
    }
}

```

```

SQLINTEGER errNo;
SQLSMALLINT msgLength;
SQLTCHAR errMsg[MSG_LEN];

if (SQL_SUCCESS == SQLError ( henv, NULL, NULL, NULL, &errNo,
errMsg, MSG_LEN, &msgLength ))
{
printf(" Error : # %lld, %s\n", errNo, errMsg);
}
exit(1);
}

/* Establish the connection */
sprintf((char*)constr,
"DSN=%s", DSN);

if ( SQLDriverConnect(hdbc, NULL, constr, SQL_NTS, NULL, 0, NULL,
SQL_DRIVER_COMPLETE))
{
printf("DBNAME = %s\n", DBNAME);
printf("USERNAME = %s\n", USERNAME);
printf("Connection error!!\n");
SQLINTEGER errNo;
SQLSMALLINT msgLength;
SQLTCHAR errMsg[MSG_LEN];

if (SQL_SUCCESS == SQLError ( henv, hdbc, NULL, NULL, &errNo,
errMsg, MSG_LEN, &msgLength ))
{
printf(" Error : # %lld, %s\n", errNo, errMsg);
}

SQLFreeConnect(hdbc);
SQLFreeEnv(henv);
exit(1);
}
printf("connected...\n");

/* Allocate memory for the statement */
if ( SQLAllocStmt(hdbc, &hstmt) == SQL_ERROR )
{
printf("AllocStmt error!!\n");
SQLDisconnect(hdbc);
SQLFreeConnect(hdbc);
SQLFreeEnv(henv);
exit(1);
}

/* Execute the query */
sprintf(query,"DROP TABLE TEST001");
SQLExecDirect(hstmt,(SQLTCHAR*)query, SQL_NTS);

```



```

sprintf(query,"CREATE TABLE TEST001 ( name varchar(20), age number(3) );");
if (SQL_ERROR == SQLExecDirect(hstmt,(SQLTCHAR*)query, SQL_NTS))
{
    execute_err(hstmt, query);
}

/* Prepare the statement and bind the variable */
sprintf(query,"INSERT INTO TEST001 VALUES( ?, ? );");
if (SQL_ERROR == SQLPrepare(hstmt, (SQLTCHAR*)query, SQL_NTS))
{
    execute_err(hstmt, query);
}

if (SQL_ERROR == SQLBindParameter(hstmt, 1, SQL_PARAM_INPUT,
SQL_C_CHAR, SQL_CHAR, 0, 0, name,
19, &len))
{
    printf("SQLBindParameter error!!! ==> %s \n",query);
    exit(1);
}

if (SQL_ERROR == SQLBindParameter(hstmt, 2, SQL_PARAM_INPUT,
SQL_C_SLONG, SQL_NUMERIC, 0, 0,
&age, 0, &len))
{
    printf("SQLBindParameter error!!! ==> %s \n",query);
    exit(1);
}

/* Execute the prepared statement */
sprintf(name, "John");
age = 28;
if (SQL_ERROR == SQLExecute(hstmt))
{
    execute_err(hstmt, query);
}

sprintf(name, "Mike");
age = 25;
if (SQL_ERROR == SQLExecute(hstmt))
{
    execute_err(hstmt, query);
}

sprintf(name, "Jessica");
age = 34;
if (SQL_ERROR == SQLExecute(hstmt))
{
    execute_err(hstmt, query);
}

sprintf(query,"SELECT * FROM TEST001");

```

```

if (SQL_ERROR == SQLExecDirect(hstmt,(SQLTCHAR*)query, SQL_NTS))
{
    execute_err(hstmt, query);
}

/* Store the SELECT result value to the variable */
if (SQL_ERROR == SQLBindCol(hstmt, 1, SQL_C_CHAR, name, 21, &len))
{
    printf("SQLBindCol error!!!\n");
    exit(1);
}

if (SQL_ERROR == SQLBindCol(hstmt, 2, SQL_C_SLONG,&age, 0, &len))
{
    printf("SQLBindCol error!!!\n");
    exit(1);
}

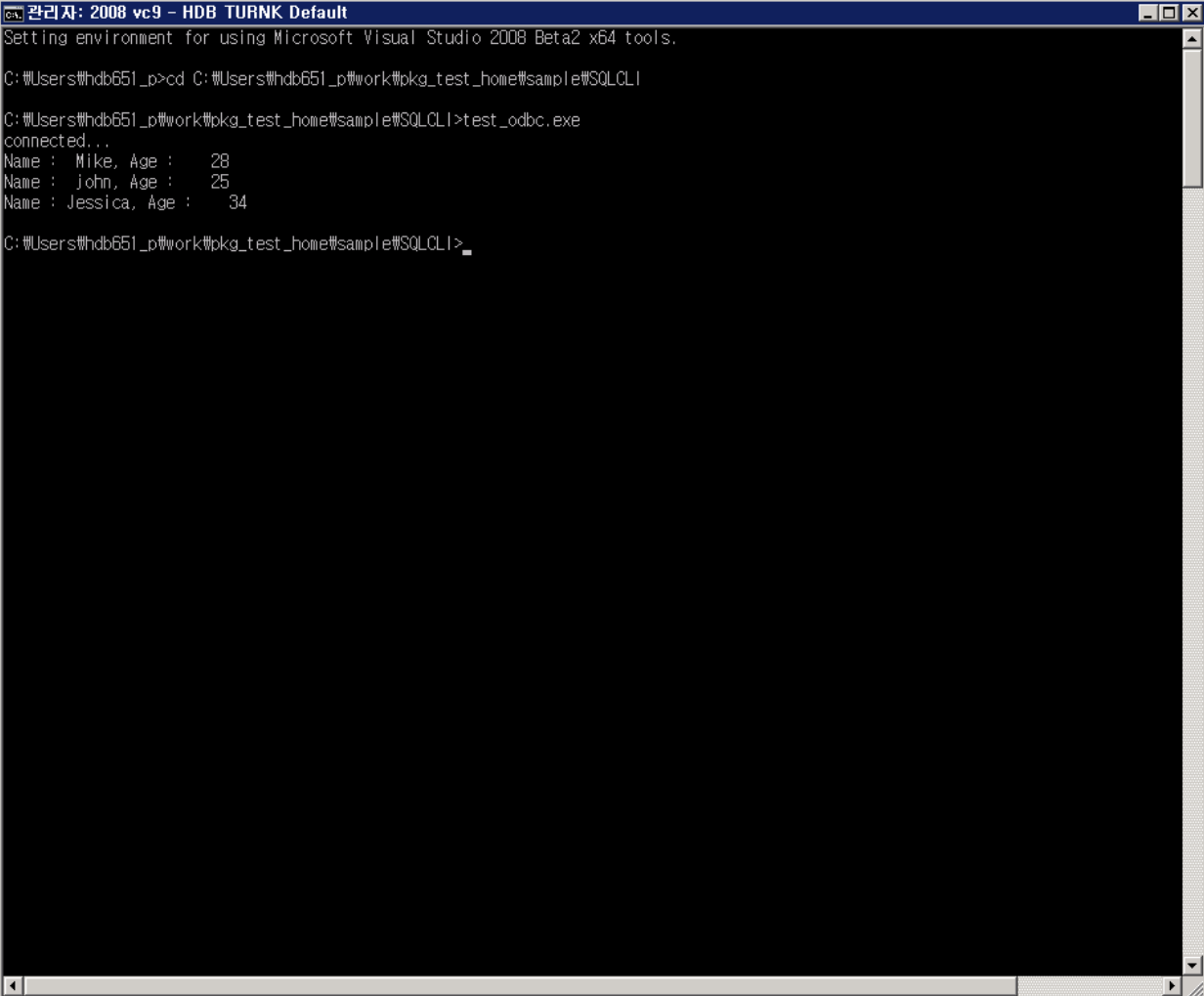
while ( SQLFetch(hstmt) == SQL_SUCCESS)
// Print the result vaule while the result value is on the screen */
{
    printf("Name : %5s, Age : %5ld\n",name,age);
}

/* Release all the handles and terminate the connection */
SQLFreeStmt(hstmt, SQL_DROP);
SQLDisconnect(hdbc);
SQLFreeConnect(hdbc);
SQLFreeEnv(henv);
}

```

## Execution Result

If you run the exe file generated after compiling in Visual C++, you will get the following result.



```
관리자: 2008 vc9 - HDB TURNK Default
Setting environment for using Microsoft Visual Studio 2008 Beta2 x64 tools.

C:\Users\hdb651_p>cd C:\Users\hdb651_p\work\pkg_test_home\sample\SQLCLI

C:\Users\hdb651_p\work\pkg_test_home\sample\SQLCLI>test_odbc.exe
connected...
Name : Mike, Age : 28
Name : John, Age : 25
Name : Jessica, Age : 34

C:\Users\hdb651_p\work\pkg_test_home\sample\SQLCLI>
```

## Example of Using LOB

This section describes how to handle LOB data using the Altibase ODBC driver with examples..

The Altibase LOB Locator requires that LOB data is handled in a session in non-autocommit mode. For more detailed information, please refer to Chapter 3. LOB Interface in the Altibase CLI User's Manual.

The user must also set the LongDataCompat property to ON in the connection string as follows:

```
"DSN=ALTIBASE;LongDataCompat=ON"
```

Or

```
"DRIVER=ALTIBASE_HDB_ODBC_64bit;User=SYS;Password=xxx;Server=127.0.0.1;PORT=20300;NL
S_USE=US7ASCII;LongDataCompat=ON"
```

The following is an example of inserting/selecting BLOB data into/from a table in C#.

```
FileStream fs = new FileStream("c:\\test.dat", FileMode.Open, FileAccess.Read);
Byte[] blob = new byte[fs.Length];
fs.Read(blob, 0, System.Convert.ToInt32(fs.Length));
fs.Close();
```

```

OdbcTransaction tx = cn.BeginTransaction();
cmd.Transaction = tx;

cmd.CommandText = "INSERT INTO T1 (C1, C2) VALUES (?, ?)";
cmd.Parameters.Add("C1", OdbcType.Int);
cmd.Parameters.Add("C2", OdbcType.Binary);

cmd.Parameters[0].Value = 1;
cmd.Parameters[1].Value = blob;

cmd.ExecuteNonQuery();
tx.Commit();

// BLOB SELECT
cmd.CommandText = "SELECT binary_length(C2), C2 FROM T1";

tx = cn.BeginTransaction();
cmd.Transaction = tx;
OdbcDataReader dr = cmd.ExecuteReader();
int len;

while (dr.Read())
{
    len = dr.GetInt32(0);
    Byte[] ff = new Byte[len];
    dr.GetBytes(1, 0, ff, 0, len);

    fs = new FileStream("c:\\test.dat", FileMode.CreateNew, FileAccess.Write);
    fs.Write(ff, 0, len);
    fs.Close();
}

```