

Altibase Tools & Utilities

# iSQL User's Manual

Release 6.1.1

May 11, 2012



---

Altibase Tools & Utilities iSQL User's Manual

Release 6.1.1

Copyright © 2001~2012 Altibase Corporation. 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 Corporation

10F, Daerung PostTower II, 182-13,

Guro-dong Guro-gu Seoul, 152-847, Korea

Telephone: +82-2-2082-1000      Fax: 82-2-2082-1099

E-mail: [support@altibase.com](mailto:support@altibase.com)      [www: http://www.altibase.com](http://www.altibase.com)

---

# Contents

<b>Preface</b>	<b>i</b>
About This Manual	ii
Intended Audience	ii
Software Environment	ii
Organization	ii
Documentation Conventions	ii
Related Documents	v
On-line Manuals	v
Altibase Welcomes Your Comments	v
<b>1. Using iSQL</b>	<b>1</b>
1.1 iSQL Overview	2
1.1.1 iSQL Main Functionality	2
1.2 Setting Up iSQL	3
1.3 iSQL Command-Line Options	5
1.4 iSQL Commands	8
1.5 iSQL Environment Variables	14
1.5.1 ALTIBASE_HOME	14
1.5.2 ALTIBASE_PORT_NO	14
1.5.3 ALTIBASE_NLS_USE	14
1.5.4 ALTIBASE_NLS_NCHAR_LITERAL_REPLACE	15
1.5.5 ISQL_CONNECTION	15
1.5.6 ISQL_BUFFER_SIZE	15
1.5.7 ALTIBASE_DATE_FORMAT	15
1.5.8 ISQL_EDITOR	16
1.5.9 ALTIBASE_IPC_FILEPATH	16
1.6 Personalizing iSQL	17
1.6.1 glogin.sql	17
1.6.2 login.sql	17
1.6.3 Editing the LOGIN file	17
<b>2. Examples of iSQL in Use</b>	<b>19</b>
2.1 Logging In to iSQL	20
2.1.1 Login Restrictions	20
2.2 Starting Up and Shutting Down ALTIBASE HDB	21
2.2.1 Starting Up ALTIBASE HDB	21
2.2.2 Shutting Down ALTIBASE HDB	21
2.3 Connecting and Disconnecting	23
2.3.1 Connecting to a Database	23
2.3.2 Disconnecting from a Database	26
2.4 Retrieving Information Related to the Database and Database Objects	27
2.4.1 Performance Views	27
2.4.2 Viewing the List of Tables	27
2.4.3 Viewing a Table Structure	28
2.4.4 Viewing Sequence Information	28
2.5 Controlling Transactions	30
2.5.1 Defining Transaction Modes	30
2.5.2 PLANCOMMIT	30
2.6 File Management	31
2.6.1 Saving Results	31
2.6.2 Running Scripts	31
2.6.3 Saving SQL Statements	32
2.6.4 Loading SQL Statements	33
2.6.5 Saving DML Statements	33
2.6.6 Editing Query Statements	33
2.7 Formatting SELECT Query Results	36
2.7.1 SET LINESIZE	36

2.7.2 SET LOBSIZE .....	36
2.7.3 SET LOBOFFSET .....	37
2.7.4 SET FEEDBACK .....	37
2.7.5 SET PAGESIZE .....	38
2.7.6 SET HEADING.....	38
2.7.7 SET COLSIZE.....	39
2.8 Setting Output Options .....	40
2.8.1 Getting the Elapsed Time .....	40
2.8.2 Setting Execution Time Units for Output.....	40
2.8.3 Describing Foreign Key Information .....	41
2.8.4 Outputting Script Execution Results .....	42
2.8.5 Outputting an Execution Plan .....	43
2.8.6 Setting Result Output Orientation .....	45
2.9 Viewing iSQL Display Settings .....	46
2.10 Host Variables .....	48
2.10.1 Declaring a Host Variable .....	48
2.10.2 Assigning a Value to a Host Variable.....	48
2.10.3 Viewing Host Variables .....	49
2.11 Executing Prepared SQL Statements.....	50
2.11.1 Prepared SQL versus Dynamic SQL Statements .....	50
2.11.2 Prepared SQL Statements .....	50
2.12 Creating, Executing and Dropping Stored Procedures.....	51
2.12.1 Creating Procedures .....	51
2.12.2 Executing Procedures.....	51
2.12.3 Dropping Procedures .....	54
2.13 Creating, Executing and Dropping Functions.....	56
2.13.1 Creating Functions .....	56
2.13.2 Executing Functions.....	56
2.13.3 Dropping Functions .....	57
2.14 Convenient User Functions.....	58
2.14.1 History .....	58
2.14.2 Shell Commands .....	58
2.14.3 Getting Help.....	58
2.15 Using National Character Sets.....	60

# Preface

---

# About This Manual

This manual describes how to use iSQL to access a database.

## Intended Audience

The following ALTIBASE® HDB™ users will find this manual useful:

- database administrators
- performance managers
- database administrators
- application developers
- technical support workers

It is recommended that 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 database server, operating system or network administration

## Software Environment

This manual has been prepared assuming that ALTIBASE HDB 5.5.1 is used as the database server.

## Organization

This manual is organized as follows:

- [Chapter1: Using iSQL](#)

This chapter presents an overview of iSQL and explains the commands and how to use iSQL.

- [Chapter2: Examples of iSQL in Use](#)

This chapter provides in-depth examples of each of the commands provided with iSQL.

## Documentation Conventions






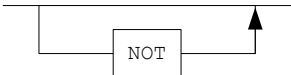
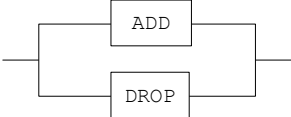
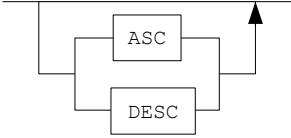
This section describes the conventions used in this manual. Understanding these conventions will make it easier to find information in this manual and other manuals in the series.

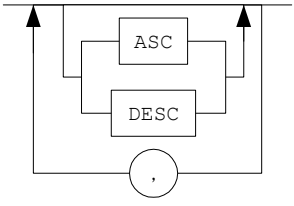
There are two sets of conventions:

- syntax diagrams
- sample code conventions

## Syntax Diagrams

This manual describes command syntax using diagrams composed of the following elements:

Elements	Meaning
	The start of a command. If a syntactic element starts with an arrow, it is not a complete command.
	The command continues to the next line. If a syntactic element ends with this symbol, it is not a complete command.
	The command continues from the previous line. If a syntactic element starts with this symbol, it is not a complete command.
	The end of a statement.
	Indicates a mandatory element.
	Indicates an optional element.
	Indicates a mandatory element comprised of options. One, and only one, option must be specified.
	Indicates an optional element comprised of options.

Elements	Meaning
 <p>The diagram shows a horizontal line with two upward-pointing arrows. Between these arrows are two boxes labeled 'ASC' and 'DESC'. Below the line, a circle contains a comma ','. Lines connect the arrows to the boxes and the circle, indicating a choice between the two options and that the entire construct is optional.</p>	Indicates an optional element in which multiple elements may be specified. A comma must precede all but the first option.

## Sample Code Conventions

The code examples explain SQL, stored procedures, iSQL, and other command line statements.

The following table describes the printing conventions used in the code examples.

Rules	Meaning	Example
[ ]	Indicates an optional item.	<code>VARCHAR [(size)] [[FIXED  ] VARIABLE]</code>
{ }	Indicates a mandatory field for which one or more items must be selected.	<code>{ ENABLE   DISABLE   COMPILE }</code>
	A delimiter between optional or mandatory arguments.	<code>{ ENABLE   DISABLE   COMPILE }</code> <code>[ ENABLE   DISABLE   COMPILE ]</code>
. . .	Indicates that the previous argument is repeated, or that sample code has been omitted.	<code>iSQL&gt; select e_lastname from employees;</code> <code>E_LASTNAME</code> ----- Moon Davenport Kobain . . . 20 rows selected.
Other Symbols	Symbols other than those shown above are part of the actual code.	<code>EXEC :p1 := 1;</code> <code>acc NUMBER(11,2);</code>
Italics	Statement elements in italics indicate variables and special values specified by the user.	<code>SELECT * FROM table_name;</code> <code>CONNECT userID/password;</code>
Lower Case Characters	Indicate program elements set by the user, such as table names, column names, file names, etc.	<code>SELECT e_lastname FROM employees;</code>



Rules	Meaning	Example
Upper Case Characters	Keywords and all elements provided by the system appear in upper case.	DESC SYSTEM_.SYS_INDICES_;

## Related Documents

For more detailed information, please refer to the following documents:

- ALTIBASE HDB Installation Guide
- ALTIBASE HDB Getting Started
- ALTIBASE HDB Administrators' Manual
- ALTIBASE HDB Replication Manual
- ALTIBASE HDB Precompiler Users' Manual
- ALTIBASE HDB ODBC Reference
- ALTIBASE HDB Application Program Interface Users' Manual
- ALTIBASE HDB iSQL Users' Manual
- ALTIBASE HDB Utilities Manual
- ALTIBASE HDB Error Message Reference

## On-line Manuals

Manuals are available at Altibase Technical Center (<http://atc.altibase.com/>).

## 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.

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

Write to us at the following electronic mail address : [support@altibase.com](mailto:support@altibase.com)

When you need an immediate assistance regarding technical issues, please contact Altibase Technical Center.

## About This Manual

Thank you. We appreciate your feedback and suggestions.

# 1 Using iSQL

---

## 1.1 iSQL Overview

iSQL is a user tool for accessing an Altibase database and retrieving and modifying stored data using SQL statements and a number of additional commands.

### 1.1.1 iSQL Main Functionality

#### **ALTIBASE HDB Startup and Shutdown**

iSQL allows you to perform database management tasks, such as starting up and shutting down the server, and execute SQL statements using the same command prompt.

#### **Database Connection & Disconnection**

After ALTIBASE HDB starts up, you can use various user names to connect to and disconnect from the database.

#### **Database Object Information Inquiry**

iSQL allows you to use SQL statements to query all database object information, and supports convenient commands for inquiring about main objects.

#### **Database Management via SQL Statements**

Because iSQL can be used to execute any kind of SQL statement, you can control transactions and alter databases quickly and conveniently.

#### **Functions to Improve User Convenience**

The above tasks can be easily and conveniently accomplished using the file management and editing functions, the ability to execute shell commands over iSQL, and the HISTORY function.

## 1.2 Setting Up iSQL

In order for iSQL to access a server, the following information is necessary.

- `ALTIBASE_HOME`  
A path to a server or client installation
- `server_name`  
The name (or IP address) of a computer on which ALTIBASE HDB Server is running
- `port_no`  
The port number to be used when connecting via TCP or IPC
- `user_id`  
A User ID registered in the database
- `password`  
The password corresponding to the User ID
- `NLS_USE`  
The character set with which to display retrieved data to the user

`ALTIBASE_HOME` can only be set using an environment variable, while the other settings may be made using command-line options. (For more information, please refer to [1.3 iSQL Command-Line Options](#).)

The `ALTIBASE_HOME` environment variable must be set in order to use iSQL. In the case of Windows, this is set automatically when the server is installed, but, in the case of the client, must be set manually by the user. We strongly suggest that you verify that this setting has been properly made, as the application may not run properly if this setting is not made.

`port_no` and `NLS_USE` can be set using the environment variables or the server settings file (`altibase.properties`). If these settings are made via all three methods, they will take priority as follows, in descending order:

1. command-line options
2. environment variables (`ALTIBASE_PORT_NO`, `ALTIBASE_NLS_USE`)
3. server settings file (`altibase.properties`)

Therefore, when it is desired to connect using options other than those that have been previously set, the command-line options can be used, so that it is not necessary to change the settings in the server setting file or the environment variables.

If any options have not been set, when iSQL is executed for the first time, the user will be prompted to enter the corresponding variables. At this time, it is essential to enter values that are valid and follow the proper format, otherwise iSQL may not run properly.

However, if the `NLS_USE` option in particular has not been set, no command prompt will appear at the time of execution. Instead, `US7ASCII` will be used, and a connection attempt will be made. In this case, if the character set of the database is not `US7ASCII`, the application will not execute properly, or some of the user's data may become corrupted. Thus it is paramount that `NLS_USE` be set to a suitable value for the usage environment.

In order to ensure stable iSQL operation, we recommend that the following environment variables

## 1.2 Setting Up iSQL

be set:

- `ALTIBASE_HOME` : the path to a server or client installation
- `ALTIBASE_PORT_NO` : the port number to use to connect to the server
- `ALTIBASE_NLS_USE` : the character set to use to display retrieved data to the user
- `PATH` : the path containing the executable file, which must equal `$ALTIBASE_HOME/bin`

## 1.3 iSQL Command-Line Options

The ALTIBASE HDB server must be started before iSQL is executed. The following options are case-insensitive.

```
iSQL [-H]
      [-S server_name] [-U user_id] [-P password]
      [-PORT port_no]
      [-UNIXDOMAIN_FILEPATH filepath]
      [-IPC_FILEPATH filepath]
      [-SILENT]
      [-F infile_name] [-O outfile_name] [-NLS_USE]
      [-NLS_NCHAR_LITERAL_REPLACE 0|1]
      [-prefer_ipv6]
```

- **-S *server\_name***

Specifies the name (or IP address) of a computer on which ALTIBASE HDB Server is running.

When connecting in IPC, if this option is specified, it is ignored and no error is raised regardless of the operating system. It can be a host name, an IPv4 address, or an IPv6 address. An IPv6 address must be enclosed by a left square bracket([) and a right square bracket(]). For example, in the case of localhost (meaning this computer), `localhost` can be specified as the host name, `127.0.0.1` as the IPv4 address, or `[::1]` as the IPv6 address. For more information about the IPv6 address notation, please refer to the *ALTIBASE HDB Administrator's Manual*.

- **-U *user\_id***

Specifies a user ID registered in the database.

- **-P *password***

Specifies the password corresponding to the user ID.

- **-PORT *port\_no***

Specifies the port number for connecting via TCP/IP or IPC. However, when connecting in a Unix environment via IPC, this option can be omitted. If it is specified, it is ignored and no error is raised. To connect via TCP, first set 'ISQL\_CONNECTION=TCP' on the client and then enter the PORT\_NO.

To connect via IPC in a Windows environment, set the environment variable 'ISQL\_CONNECTION=IPC' and specify the port number using one of the following:

- the -PORT option
- the ALTIBASE\_IPC\_PORT\_NO environment variable
- the IPC\_PORT\_NO property in altibase.properties

If the environment variable ISQL\_CONNECTION is not set to IPC and the -PORT option is omitted, the port number will be checked for first in the environment variable ALTIBASE\_PORT\_NO and then in the PORT\_NO property in altibase.properties, and if it is not set in either of those places, a prompt to enter it will be raised.

- **-UNIXDOMAIN\_FILEPATH *filepath***

## 1.3 iSQL Command-Line Options

When a server and client connect using a Unix domain socket in a Unix environment (ISQL\_CONNECTION=UNIX), the connection will fail if the server and client have different values for ALTIBASE\_HOME and also have different Unix domain socket paths. In this case, if the server and client use corresponding files (e.g. ALTIBASE\_HOME/trc/cm-unix), Unix domain communication is possible.

- -IPC\_FILEPATH *filepath*

When the client and the server are to connect via IPC (ISQL\_CONNECTION=IPC) in a Unix environment, if ALTIBASE\_HOME is set differently on them, they will not be able to connect if they have different socket paths. In this case, Unix domain communication can be achieved using the ALTIBASE\_HOME/trc/cm-ipc file, and then information about shared memory can be retrieved. However, this option can be omitted if ALTIBASE\_IPC\_FILEPATH is set.

- -F *infile\_name*

Specifies a script file to be executed immediately after iSQL is launched.

- -O *outfile\_name*

Specifies a file in which to store the results of the executed iSQL commands. This file will be created in the current directory. If the file already exists, it will be overwritten.

- -H

Outputs help information for iSQL execution.

- -SILENT

This option turns on silent mode. If silent mode is on, noncritical messages, such as the copyright notice, etc. will not be displayed.

- -NLS\_USE

Specifies the character set with which to display data to the user. The following character sets may be specified:

- US7ASCII
- KO16KSC5601
- MS949
- BIG5
- GB231280
- UTF8
- SHIFTJIS
- EUCJP

If omitted, the environment variable ALTIBASE\_NLS\_USE or altibase.properties will be used, in descending order of preference, and if it is still not specified, the basic character set (US7ASCII) will be used.

- -NLS\_NCHAR\_LITERAL\_REPLACE

- 0: convert all strings to the database character set without checking for the "N" character
- 1: do not convert strings that are preceded by the "N" character to the database character set



- `-prefer_ipv6`

This option determines the IP address to be connected first when a host name is given for the `-s` option.

If this option is specified and a host name is given for the `-s` option, this means that resolving the host name to the IPv6 address is preferred.

If this option is omitted, `isql` connects to the IPv4 address by default.

If it fails to connect to the preferred IP version address, an attempt is made to connect using the other IP version address.

For example, when `localhost` is given for the `-s` option and this option is specified, `isql` first tries to connect to the `:::1` IPv6 address. If this attempt fails, `isql` proceeds to connect to the `127.0.0.1` IPv4 address.

If any of the `-S`, `-U`, or `-P` options are missing from the above command, the user will be prompted to input the option values.

## 1.4 iSQL Commands

When iSQL is started, an iSQL command prompt will appear, and when iSQL commands are entered, the results of execution will be displayed. The iSQL commands are described individually in the following table.

Category	Type	Commands	Description
iSQL startup and shut-down	Startup	\$ isql [option]	If you execute this command in a shell, iSQL will start up. For information on the available options, please refer to the <a href="#">1.3 iSQL Command-Line Options</a> section.
	Prompt	iSQL>	Type a command at the iSQL prompt and press the ENTER key.
	Shutdown	EXIT; QUIT;	Used to shut down iSQL.
ALTIBASE HDB startup and shutdown	ALTIBASE HDB Startup	STARTUP	Use the PRE-PROCESS, PROCESS, CONTROL, META, or SERVICE option to start ALTIBASE HDB up to the corresponding stage.
	ALTIBASE HDB Shutdown	SHUTDOWN	Use one of the NORMAL, IMMEDIATE, or ABORT options to shut down ALTIBASE HDB.
Database connection and disconnection	Access the server as another user	CONNECT [ <i>logon</i> ] [ <i>nls</i> ] [AS sysdba]; where <i>logon</i> has the syntax: <i>user1</i> [/ <i>pass1</i> ] where <i>nls</i> has the syntax: NLS= <i>character_set</i>	This command allows access to the database as <i>user1</i> with password <i>pass1</i> after having already accessed the database as another user in iSQL. If CONNECT is successful, the information related to the previous session is cleared. The AS clause allows the SYS user to access the server in sysdba manager mode. Only one user is allowed to connect as sysdba at a time. The <i>nls</i> option specifies the character set. For detailed information on character sets, please refer to the <a href="#">1.3 iSQL Command-Line Options</a> : -NLS_USE option.
	Terminate a connection	DISCONNECT ;	Ends the current session and terminates the connection with the server.

Category	Type	Commands	Description
Database object information inquiry	Display performance view list	SELECT * FROM V\$TAB;	Displays the list of all of the performance views provided by the system. This command is available only in iSQL.
	Display table list	SELECT * FROM TAB ;	Displays the list of currently created tables. This command is only available in iSQL.
	Display table structure	DESC samp ;	Lists the column definitions for the table <i>samp</i>
	Display sequence Information	SELECT * FROM V\$SEQ;	If you accessed the server with the SYS account, information on all sequences is displayed. If you accessed the server as another user, only the information on the sequences generated by that user will be displayed. This command is available only in iSQL.
Transaction control	Setting transaction mode	AUTOCOMMIT ON; AUTOCOMMIT OFF;	Determines whether to commit commands automatically at the time that they are executed. Default: ON
	Other SET functions	SET PLANCOMMIT ON; SET PLANCOMMIT OFF;	Determines whether to automatically commit commands such as DESC, SELECT * FROM TAB, or SELECT * FROM <i>seq_name</i> when EXPLAIN PLAN is ON (or ONLY) and AUTOCOMMIT is OFF. Default: OFF

## 1.4 iSQL Commands

Category	Type	Commands	Description
File management	Output data to a file	SPOOL file_name ;	Starts writing the results shown on the screen to the file <i>file_name</i> .
		SPOOL OFF;	Stops writing the results shown on the screen to the file <i>file_name</i> .
	SQL script execution	START file_name ;	Reads a script file and executes the SQL statements in sequence.
		@ file_name;	Performs a function similar to that of startup when executed via an iSQL prompt.
		@@ file_name;	When used in a script, this command executes the file <i>file_name</i> in the same directory as the calling script.
	Save SQL statement to file	SAVE abc.sql;	Saves the last of the commands currently in the iSQL buffer to a file.
	Load SQL statement	LOAD abc.sql;	Loads the first of the commands saved in a file at the end of the command buffer.
	Save DML statements to file	SET QUERYLOGGING ON; SET QUERYLOGGING OFF;	This writes executed DML statements, such as INSERT, UPDATE, DELETE and MOVE, in \$ALTIBASE_HOME/trc/isql_query.log.
	Edit query statements	ED	For creating and editing temporary files.
		ED file_name[.sql]	For editing existing files or creating new files.
		2ED or 2 ED	Edits query command number 2 in the history list.

Category	Type	Commands	Description
Control output option	Format SELECT result column	SET LINESIZE 100;	Sets the length of a display line for outputting the result of a SELECT query. Must be between 10 and 32767 inclusive. Default: 80
	Format SELECT result column of type CLOB	SET LOBSIZE 10;	Sets the number of characters to display when a CLOB column is output. Default : 80
		SET LOBOFFSET 3;	Sets the number of characters by which to offset the display when a CLOB column is output. Default : 0
	Output SELECT result count	SET FEED[BACK] ON; SET FEED[BACK] OFF; SET FEED[BACK] n;	Determines whether to output the number of rows in a query result.
	Format rows of SELECT result	SET PAGESIZE 10;	Sets how many records of a SELECT query result are output at one time. When set to 0, all resultant records are output. Default: 0
	Show/hide SELECT result header	SET HEADING ON; SET HEADING OFF;	Sets whether to output the header of a SELECT result Default: ON
	Set SELECT result output size.	SET COLSIZE N ;	Sets the number of characters to output when CHAR or VARCHAR type columns are output as a SELECT query result.
	Show SQL statement execution time	SET TIMING ON; SET TIMING OFF;	Sets whether to output the amount of time taken to execute a SQL command. Default: OFF
	Set the SQL statement execution time units for output	SET TIMESCALE SEC; SET TIMESCALE MILSEC; SET TIMESCALE MICSEC; SET TIMESCALE NANSEC;	Sets the unit of time for executing SQL statements as seconds, milliseconds, microseconds or nanoseconds.
	Show/hide foreign key information	SET FOREIGNKEYS ON; SET FOREIGNKEYS OFF;	Determines whether to include foreign key information in the output when displaying the table structure (using DESC). Default: OFF
	Show/hide script execution result	SET TERM ON; SET TERM OFF;	Determines whether to display the results of execution of a script file on the screen. Default: ON

## 1.4 iSQL Commands

Category	Type	Commands	Description
Control output option	Output executionplan tree	ALTER SESSION SET EXPLAIN PLAN = ON; ALTER SESSION SET EXPLAIN PLAN = ONLY; ALTER SESSION SET EXPLAIN PLAN = OFF;	Determines whether to output an execution plan for a SELECT statement. Default: OFF
	SELECT result output direction	SET VERTICAL ON; SET VERTICAL OFF;	Displays SELECT results vertically when set to ON. Default: OFF
	Show value of iSQL display settings	SHOW LINESIZE	Displays the current LINESIZE value.
		SHOW COLSIZE	Displays the current COLSIZE value.
		SHOW LOBOFFSET	Displays the current LOBOFFSET value.
		SHOW LOBSIZE	Displays the current LOBSIZE value.
		SHOW PAGESIZE	Displays the current PAGESIZE value.
		SHOW PLANCOMMIT	Shows whether PLANCOMMIT is ON or OFF.
		SHOW QUERYLOGGING	Shows whether DML statements will be written to ALTIBASE_HOME/trc/isql_query.log when executed.
		SHOW FEEDBACK	Shows the current FEEDBACK value.
		SHOW HEADING	Shows the current HEADING setting.
		SHOW TERM	Shows the current TERM setting.
		SHOW TIMING	Shows the current TIMING setting.
		SHOW TIMESCALE	This shows the current time units for the execution of SQL statements.
		SHOW USER	Shows the current user.
		SHOW FOREIGNKEYS	Shows the current foreign key display setting.
		SHOW VERTICAL	Shows whether the results of a SELECT query will be output vertically.
		SHOW ALL	Shows the set values of the display settings for the current session.

Category	Type	Commands	Description
Variable and Prepared SQL statements	Variable declaration	VAR <i>p1</i> INTEGER;	Declares the variable <i>p1</i> as integer type.
		VARIABLE <i>p2</i> CHAR(10);	Declares the variable <i>p2</i> as CHAR type.
	Assign values to variables	EXECUTE : <i>p1</i> := 100 ;	Assigns the value 100 to variable <i>p1</i> .
		EXEC : <i>p2</i> := 'abc' ;	Assigns the text 'abc' to variable <i>p2</i> .
	Variable display	PRINT VAR[ <i>IABLE</i> ] ;	Shows the currently declared variables.
		PRINT <i>p1</i> ;	Shows the type and value of variable <i>p1</i> .
	Prepared SQL statement execution	PREPARE SQL statement ;	Separates the processes of query optimization and execution, and executes the query as a prepared SQL statement. In iSQL, the default execution method for executing SQL statements is the Direct Execution method, in which optimization and execution are performed at once. There is no difference between the two execution methods in iSQL in terms of the results obtained, however, prepared SQL statements can be used to bind variables to values and execute SQL statements based thereon.
Functions for user convenience	Historylist display	HISTORY; H;	Shows a list of the commands currently saved in the iSQL buffer.
	Repeat execution	/	Repeats execution of the command currently in the iSQL buffer. The most recently executed command will be executed again.
		2/	Executes the second command in a list output using the HISTORY command.
	Shell command execution	! shell command	A shell command that follows an exclamation point will be immediately executed from within iSQL.
	Comment	/* comment */ -- comment	Indicate a multiple-line comment and a single-line comment, respectively.
	Help	HELP; HELP INDEX; HELP EXIT;	Provides information on how to use help, outputs a list of commands, and describes (e.g.) the EXIT command, respectively.

# 1.5 iSQL Environment Variables

### 1.5.1 ALTIBASE\_HOME

Sets the directory in which the package is installed.

In the case of MS Windows, this is set automatically when the server is installed, however, when the client is installed, this is not automatically set due to the danger of a conflict with the environment variables for the server. When installing the client, the user must manually set this directory.

This environmental variable must be set in order to use iSQL.

Ex)

```
Windows (Server): 'set ALTIBASE_HOME=C:/Program Files/Altibase/  
Altibase5_Server/altibase_home',
```

```
Windows (Client): 'set ALTIBASE_HOME=C:/Program Files/Altibase/  
Altibase5_Client/altibase_home_client')
```

### 1.5.2 ALTIBASE\_PORT\_NO

This is the port number of the server to connect to. This can be specified either by using the -PORT option or in altibase.properties.

If no designated port number can be found (in descending order of precedence) in the -PORT option, in the environment variable ALTIBASE\_PORT\_NO, or in altibase.properties, a prompt to enter the port number will appear.

### 1.5.3 ALTIBASE\_NLS\_USE

This is the character set used to display retrieved results to the user.

- US7ASCII
- KO16KSC5601
- MS949
- BIG5
- GB231280
- UTF8
- SHIFTJIS
- EUCJP

This can be set either using the -NLS\_USE option or in altibase.properties.

If NLS\_USE is not specified using the -NLS\_USE option, the environment variable ALTIBASE\_NLS\_USE, or altibase.properties (in descending order of precedence), US7ASCII is used as



the default character set.

### 1.5.4 ALTIBASE\_NLS\_NCHAR\_LITERAL\_REPLACE

By default, iSQL converts an entire query string to the database character set before sending the data to the database. This behavior can be prevented for a given string literal by setting this property to 1 and placing the "N" character in front of the string literal.

A property setting of 1 instructs iSQL to search for the "N" character in front of every string literal. If the "N" character is found, iSQL sends the string to the database without converting it to the database character set. This is useful when it is desired to use NCHAR type data that are encoded differently from the database character set.

- 0: convert all strings to the database character set without checking for the "N" character
- 1: do not convert strings that are preceded by the "N" character to the database character set

*Note: Setting this variable to 1 can be expensive in terms of usage of client resources.*

### 1.5.5 ISQL\_CONNECTION

When ALTIBASE HDB is used in a client-server arrangement, the user can set environment variables to select the client-server protocol that is suitable for the operating environment. ALTIBASE HDB supports the TCP/IP, IPC and Unix domain socket protocols. The default protocol for communication with ALTIBASE HDB servers is TCP/IP. Note that when using the IPC protocol the value of ALTIBASE HDB properties related to the IPC channel (IPC\_CHANNEL\_COUNT) must be considered.

The following example shows how to set the environment variable when using the IPC protocol:

```
CSH: setenv ISQL_CONNECTION IPC
SH: ISQL_CONNECTION=IPC; export ISQL_CONNECTION
```

### 1.5.6 ISQL\_BUFFER\_SIZE

The size of the buffer in which to store queries can be set using this environment variable.

```
Ex)
CSH: setenv ISQL_BUFFER_SIZE 128000
SH: ISQL_BUFFER_SIZE = 128000; export ISQL_BUFFER_SIZE
```

### 1.5.7 ALTIBASE\_DATE\_FORMAT

When retrieving Date type data using a SELECT statement, the environment variable ALTIBASE\_DATE\_FORMAT can be used to change the default date format, which is YYYY/MM/DD HH:MI:SS, to some other date format.

```
Ex) For Born, Korn, or Bash Shell
export ALTIBASE_DATE_FORMAT='DD-MON-YYYY'
```

## 1.5 iSQL Environment Variables

### 1.5.8 ISQL\_EDITOR

This environment variable can be used to change the default editor (/usr/bin/vi ).

Ex)

```
CSH: setenv ISQL_EDIT usr/bin/ed
```

```
SH: ISQL_EDIT=/usr/bin/ed; export ISQL_EDITOR
```

### 1.5.9 ALTIBASE\_IPC\_FILEPATH

In a Unix environment, if a client and the server have different values for ALTIBASE\_HOME, they will not be able to connect via IPC if they have different Unix domain socket paths. In this case, in order to be able to connect via IPC, it will be necessary to set the ALTIBASE\_IPC\_FILEPATH environment variable or the -IPC\_FILEPATH iSQL option to the \$ALTIBASE\_HOME/trc/cm-ipc file used by the server.

## 1.6 Personalizing iSQL

iSQL users can customize their iSQL environment and use the same settings for each session. For example, using the OS file, the user can specify a desired output format so that each query result displays the current time whenever query results are output. These files can be categorized into the following two types.

### 1.6.1 glogin.sql

For initialization tasks that must be conducted when iSQL is started, iSQL supports the creation of a global script file, `glogin.sql`, by the DB administrator. iSQL executes this script whenever any user executes iSQL or attempts to connect to ALTIBASE HDB for the first time. The global file allows the DB administrator to make site-specific iSQL environment settings for all users. The global script file is located in `$ALTIBASE_HOME/conf`.

### 1.6.2 login.sql

iSQL also supports the `login.sql` file, which is executed after `glogin.sql`. If both the `glogin.sql` file and the `login.sql` file exist, `login.sql` is executed after `glogin.sql` during iSQL startup, so the commands in `login.sql` will take precedence.

If several people share one Unix account, it will be impossible for them to personalize the `glogin.sql` file. In this case, individual users may add SQL commands, stored procedures, or iSQL commands to their respective `login.sql` files in their personal work directories. When a user starts up iSQL, iSQL automatically searches the current directory for the `login.sql` file and executes the commands in it.

The `login.sql` file cannot modify initial iSQL settings or individual session actions.

### 1.6.3 Editing the LOGIN file

The user may change the LOGIN file, like any other script. The following is an example of user1 creating a LOGIN file that turns off autocommit mode and executes SQL statements:

```
$ vi glogin.sql
AUTOCOMMIT ON
SET HEADING OFF
SELECT sysdate FROM dual;

$ vi login.sql
AUTOCOMMIT OFF
SET HEADING ON
DROP TABLE savept;
CREATE TABLE savept(num INTEGER);
INSERT INTO savept VALUES(1);
SAVEPOINT sp1;
INSERT INTO savept VALUES(2);
SELECT * FROM savept;
ROLLBACK TO SAVEPOINT sp1;
SELECT * FROM savept;
COMMIT;

$ isql
-----
Altibase Client Query utility.
```

## 1.6 Personalizing iSQL

```
Release Version 6.1.1.1
Copyright 2000, Altibase Corporation or its subsidiaries.
All Rights Reserved.
-----
Write Server Name (default:127.0.0.1) :
Write UserID : user1
Write Password :
ISQL_CONNECTION = TCP, SERVER = 127.0.0.1, PORT_NO = 20300
Set autocommit on success.    -> Executing glogin.sql first

28-DEC-2004                -> heading off
1 row selected.
Set autocommit off success.  -> Execute login.sql in the current work directory of the user after
glogin.sql is executed.
Drop success.
Create success.
1 row inserted.
Savepoint success.          -> It is executable only when Autocommit mode is off
1 row inserted.
NUM                          -> heading on
-----
1
2
2 rows selected.
Rollback success.
NUM
-----
1
1 row selected.
Commit success.
```

# **2 Examples of iSQL in Use**

---

This chapter describes several examples of the use of iSQL to manipulate databases.

# 2.1 Logging In to iSQL

To use iSQL, users must first be logged in. Connection information may be input directly via a command line, or via the iSQL input prompt.

```
isql -U userID -P password [-SYSDBA]
```

or

```
isql [-SYSDBA]
```

Additional information necessary for connection with the server is the server name (-S), user ID (-U), and password (-P). The user ID and password are not case-sensitive.

In order for the SYS user to use iSQL as an administrator, the SYSDBA option is used. The SYSDBA option can be used for remote access.

## 2.1.1 Login Restrictions

- Only one user is permitted to connect in SYSDBA mode at one time. Two or more users cannot connect in SYSDBA mode at the same time.
- You can access the database remotely in SYSDBA mode, but can't start up the database.

For detailed information on system privileges, please refer to the *ALTIBASE HDB SQL Reference*. For detailed information on errors that may arise during iSQL execution, please refer to the *ALTIBASE HDB Error Message Reference*.

```
$ isql -S 127.0.0.1 -U sys -P manager [-SYSDBA]
```

or

```
$ isql [-sysdba]
```

```
-----  
Altibase Client Query utility.
```

```
Release Version 6.1.1.1
```

```
Copyright 2000, Altibase Corporation or its subsidiaries.
```

```
All Rights Reserved.  
-----
```

```
Write Server Name (default:127.0.0.1) :
```

```
Write UserID : sys
```

```
Write Password : manager -> The password on the screen is not displayed.
```

```
ISQL_CONNECTION = UNIX, SERVER = 127.0.0.1, PORT_NO = 20300
```

```
iSQL(sysdba) >
```

```
-> iSQL is connected to the server, and SQL, iSQL, and PSM commands  
can be input and executed here.
```

## 2.2 Starting Up and Shutting Down ALTIBASE HDB

iSQL can be used to start up and shut down ALTIBASE HDB.

### 2.2.1 Starting Up ALTIBASE HDB

To start up ALTIBASE HDB, iSQL must first be launched with the `-sysdba` option, in the same way as when a database is created.

*Note: ALTIBASE HDB startup commands can be executed only with the UNIX account with which ALTIBASE HDB (including iSQL) was installed.*

The following is an example of the use of iSQL to start up ALTIBASE HDB. For more information on starting up ALTIBASE HDB, please refer to the *ALTIBASE HDB Administrators' Manual Chapter 4: Startup and Shutdown*.

```
$ isql -s 127.0.0.1 -u sys -p manager -sysdba
-----
Altibase Client Query utility.
Release Version 6.1.1.1
Copyright 2000, Altibase Corporation or its subsidiaries.
All Rights Reserved.
-----
ISQL_CONNECTION = UNIX, SERVER = 127.0.0.1, PORT_NO = 20300
[ERR-910FB : Connected to idle instance]

iSQL(sysdba)> startup service
Connecting to the DB server... Connected.

TRANSITION TO PHASE : PROCESS

TRANSITION TO PHASE : CONTROL

TRANSITION TO PHASE : META
[SM] Recovery Phase - 1 : Preparing Database
                        : Dynamic Memory Version => Parallel Loading
[SM] Recovery Phase - 2 : Loading Database
[SM] Recovery Phase - 3 : Skipping Recovery & Starting Threads...
                        Refining Disk Table
[SM] Refine Memory Table :
.....
..... [SUCCESS]
[SM] Rebuilding Indices [Total Count:101]
.....
..... [SUCCESS]
TRANSITION TO PHASE : SERVICE
[CM] Listener started : TCP on port 20300
[CM] Listener started : UNIX
[RP] Initialization : [PASS]
--- STARTUP Process SUCCESS ---
Command execute success.
```

### 2.2.2 Shutting Down ALTIBASE HDB

Use the SHUTDOWN command to shut down a running ALTIBASE HDB server.

## 2.2 Starting Up and Shutting Down ALTIBASE HDB

The following is an example of the use of iSQL to shut down ALTIBASE HDB. For more information on shutting down ALTIBASE HDB, please refer to the *ALTIBASE HDB Administrators' Manual Chapter 4: Startup and Shutdown*.

```
iSQL(sysdba)> shutdown normal  
Ok..Shutdown Proceeding....
```

```
TRANSITION TO PHASE : Shutdown Altibase  
[RP] Finalization : PASS  
shutdown normal success.
```



## 2.3 Connecting and Disconnecting

### 2.3.1 Connecting to a Database

The CONNECT command is used to connect to ALTIBASE HDB with a specified user ID. If the first connection attempt fails, the CONNECT command does not prompt again for the user ID or password.

```
CONNECT [logon] [nls] [AS SYSDBA];
```

where *logon* has the syntax:

```
userID[/password]
```

and *nls* has the syntax:

```
NLS=character_set
```

#### 2.3.1.1 userID/password

The user ID and password with which to establish a connection to ALTIBASE HDB.

#### 2.3.1.2 NLS=character\_set

The NLS option specifies the character set.

```
iSQL> CONNECT sys/manager NLS=US7ASCII
Connect success.
```

#### 2.3.1.3 AS SYSDBA

The AS clause permits the SYS user to access the server in sysdba manager mode.

If CONNECT is successful, the current session is terminated, and a connection is established to the server using the specified user ID and password and the information in altibase.properties. Accordingly, the session information is cleared before connecting.

For instance, if AUTOCOMMIT mode is set to TRUE in altibase.properties and AUTOCOMMIT mode is changed to FALSE in iSQL, when the CONNECT statement is executed, AUTOCOMMIT mode will be changed to TRUE, because of the value in altibase.properties.

If CONNECT fails, the previous session is terminated and the connection with the server is closed. In other words, the result of all SQL statements executed thereafter will be a "Not connected" message. Execute "CONNECT userID/password [AS SYSDBA]" to attempt to re-establish a connection with the server.

```
$ isql
-----
Altibase Client Query utility.
Release Version 6.1.1.1
Copyright 2000, Altibase Corporation or its subsidiaries.
All Rights Reserved.
-----
Write Server Name (default:127.0.0.1) :
```

## 2.3 Connecting and Disconnecting

```
Write UserID : SYS
Write Password :
ISQL_CONNECTION = TCP, SERVER = 127.0.0.1, PORT_NO = 20300
```

```
isQL> SHOW USER;
User : SYS
```

```
isQL> CREATE USER altiadmin IDENTIFIED BY alti1234;
Create success.
```

```
isQL> CONNECT altiadmin/alti1234;
Connect success.
```

```
isQL> SHOW USER;
User : ALTIADMIN
```

```
isQL> CREATE TABLE altitbl(i1 INTEGER, i2 CHAR(5));
Create success.
```

```
isQL> SELECT * FROM tab;
TABLE NAME                                TYPE
-----
ALTTITBL                                TABLE
CLEAR_DP                                SYNONYM
DUAL                                    SYNONYM
EXPORT_PARTITION_TO_FILE                 SYNONYM
EXPORT_TO_FILE                           SYNONYM
EXPORT_USER_TABLES                       SYNONYM
FCLOSE                                  SYNONYM
FCLOSE_ALL                              SYNONYM
FCOPY                                    SYNONYM
FFLUSH                                  SYNONYM
FOPEN                                    SYNONYM
FREMOVE                                  SYNONYM
FRENAME                                  SYNONYM
GET_LINE                                SYNONYM
IMPORT_FROM_FILE                         SYNONYM
IS_OPEN                                  SYNONYM
NEW_LINE                                SYNONYM
PRINT                                    SYNONYM
PRINTLN                                  SYNONYM
PUT                                      SYNONYM
PUT_LINE                                SYNONYM
RAISE_APPLICATION_ERROR                  SYNONYM
REGISTER                                  SYNONYM
REMOVE                                  SYNONYM
REMOVEALL                                SYNONYM
REMOVE_DP                                SYNONYM
REMOVE_XID                               SYNONYM
RESUME_DP                                SYNONYM
SET_DEFAULTS                             SYNONYM
SIGNAL                                    SYNONYM
SLEEP                                    SYNONYM
WAITANY                                  SYNONYM
WAITONE                                  SYNONYM
33 rows selected.
```

```
isQL> CONNECT sys/manager;
Connect success.
```

```
isQL> SHOW USER;
User : SYS
```

```
isQL> CREATE TABLE systbl(i1 INTEGER, i2 CHAR(5));
Create success.
```

```

isQL> SELECT * FROM tab;
USER NAME      TABLE NAME      TYPE
-----
SYSTEM_        STO_COLUMNS_      SYSTEM TABLE
SYSTEM_        STO_DATUMS_       SYSTEM TABLE
SYSTEM_        STO_ELLIPSOIDS_   SYSTEM TABLE
SYSTEM_        STO_GEOCCS_       SYSTEM TABLE
SYSTEM_        STO_GEOGCS_       SYSTEM TABLE
SYSTEM_        STO_PRIMEMS_      SYSTEM TABLE
SYSTEM_        STO_PROJCS_       SYSTEM TABLE
SYSTEM_        STO_PROJECTIONS_  SYSTEM TABLE
SYSTEM_        STO_SRS_          SYSTEM TABLE
SYSTEM_        STO_USER_COLUMNS_ SYSTEM TABLE
SYSTEM_        SYS_COLUMNS_      SYSTEM TABLE
SYSTEM_        SYS_COMMENTS_     SYSTEM TABLE
SYSTEM_        SYS_CONSTRAINTS_  SYSTEM TABLE
SYSTEM_        SYS_CONSTRAINT_COLUMNS_ SYSTEM TABLE
SYSTEM_        SYS_DATABASE_     SYSTEM TABLE
SYSTEM_        SYS_DATABASE_LINKS_ SYSTEM TABLE
SYSTEM_        SYS_DATA_PORTS_   SYSTEM TABLE
SYSTEM_        SYS_DIRECTORIES_  SYSTEM TABLE
SYSTEM_        SYS_DN_USERS_     SYSTEM TABLE
SYSTEM_        SYS_DUMMY_        SYSTEM TABLE
SYSTEM_        SYS_ENCRYPTED_COLUMNS_ SYSTEM TABLE
SYSTEM_        SYS_GRANT_OBJECT_ SYSTEM TABLE
SYSTEM_        SYS_GRANT_SYSTEM_ SYSTEM TABLE
SYSTEM_        SYS_INDEX_COLUMNS_ SYSTEM TABLE
SYSTEM_        SYS_INDEX_PARTITIONS_ SYSTEM TABLE
SYSTEM_        SYS_INDICES_      SYSTEM TABLE
SYSTEM_        SYS_LOBS_         SYSTEM TABLE
SYSTEM_        SYS_PART_INDICES_ SYSTEM TABLE
SYSTEM_        SYS_PART_KEY_COLUMNS_ SYSTEM TABLE
SYSTEM_        SYS_PART_LOBS_    SYSTEM TABLE
SYSTEM_        SYS_PART_TABLES_  SYSTEM TABLE
SYSTEM_        SYS_PRIVILEGES_   SYSTEM TABLE
SYSTEM_        SYS_PROCEDURES_   SYSTEM TABLE
SYSTEM_        SYS_PROC_PARAS_   SYSTEM TABLE
SYSTEM_        SYS_PROC_PARSE_   SYSTEM TABLE
SYSTEM_        SYS_PROC_RELATED_ SYSTEM TABLE
SYSTEM_        SYS_REPLICATIONS_ SYSTEM TABLE
SYSTEM_        SYS_REPL_HOSTS_   SYSTEM TABLE
SYSTEM_        SYS_REPL_ITEMS_   SYSTEM TABLE
SYSTEM_        SYS_REPL_OFFLINE_DIR_ SYSTEM TABLE
SYSTEM_        SYS_REPL_OLD_COLUMNS_ SYSTEM TABLE
SYSTEM_        SYS_REPL_OLD_INDEX_COLUMNS_ SYSTEM TABLE
SYSTEM_        SYS_REPL_OLD_INDICES_ SYSTEM TABLE
SYSTEM_        SYS_REPL_OLD_ITEMS_ SYSTEM TABLE
SYSTEM_        SYS_REPL_RECOVERY_INFOS_ SYSTEM TABLE
SYSTEM_        SYS_SECURITY_     SYSTEM TABLE
SYSTEM_        SYS_SYNONYMS_     SYSTEM TABLE
SYSTEM_        SYS_TABLES_       SYSTEM TABLE
SYSTEM_        SYS_TABLE_PARTITIONS_ SYSTEM TABLE
SYSTEM_        SYS_TBS_USERS_    SYSTEM TABLE
SYSTEM_        SYS_TRIGGERS_     SYSTEM TABLE
SYSTEM_        SYS_TRIGGER_DML_TABLES_ SYSTEM TABLE
SYSTEM_        SYS_TRIGGER_STRINGS_ SYSTEM TABLE
SYSTEM_        SYS_TRIGGER_UPDATE_COLUMNS_ SYSTEM TABLE
SYSTEM_        SYS_USERS_        SYSTEM TABLE
SYSTEM_        SYS_VIEWS_        SYSTEM TABLE
SYSTEM_        SYS_VIEW_PARSE_   SYSTEM TABLE
SYSTEM_        SYS_VIEW_RELATED_ SYSTEM TABLE
SYSTEM_        SYS_XA_HEURISTIC_TRANS_ SYSTEM TABLE
ALTIADMIN      ALTITBL           TABLE
SYS            SYSTBL            TABLE

```

## 2.3 Connecting and Disconnecting

CLEAR_DP	SYNONYM
DUAL	SYNONYM
EXPORT_PARTITION_TO_FILE	SYNONYM
EXPORT_TO_FILE	SYNONYM
EXPORT_USER_TABLES	SYNONYM
FCLOSE	SYNONYM
FCLOSE_ALL	SYNONYM
FCOPY	SYNONYM
FFLUSH	SYNONYM
FOPEN	SYNONYM
FREMOVE	SYNONYM
FRENAME	SYNONYM
GET_LINE	SYNONYM
IMPORT_FROM_FILE	SYNONYM
IS_OPEN	SYNONYM
NEW_LINE	SYNONYM
PRINT	SYNONYM
PRINTLN	SYNONYM
PUT	SYNONYM
PUT_LINE	SYNONYM
RAISE_APPLICATION_ERROR	SYNONYM
REGISTER	SYNONYM
REMOVE	SYNONYM
REMOVEALL	SYNONYM
REMOVE_DP	SYNONYM
REMOVE_XID	SYNONYM
RESUME_DP	SYNONYM
SET_DEFAULTS	SYNONYM
SIGNAL	SYNONYM
SLEEP	SYNONYM
WAITANY	SYNONYM
WAITONE	SYNONYM

93 rows selected.

### 2.3.2 Disconnecting from a Database

DISCONNECT is used to terminate the current session and disconnect from the server. The result of all subsequently executed SQL statements will be a “Not connected” message, and “CONNECT userID/password” must be executed in order to connect to the server again.

```
DISCONNECT;

iSQL> INSERT INTO systbl VALUES(1, 'A1');
1 row inserted.
iSQL> INSERT INTO systbl VALUES(2, 'A2');
1 row inserted.
iSQL> SELECT * FROM systbl;
I1      I2
-----
1        A1
2        A2
2 rows selected.
iSQL> DISCONNECT;
Disconnect success.
iSQL> INSERT INTO systbl VALUES(3, 'A3');
[ERR-91020 : No Connection State]
iSQL> SELECT * FROM systbl;
[ERR-91020 : No Connection State]
iSQL> CONNECT sys/manager;
Connect success.
```

## 2.4 Retrieving Information Related to the Database and Database Objects

### 2.4.1 Performance Views

A performance view is a type of data dictionary table capable of inquiring about the server status and database information. The following SELECT statement can be used to view the list of performance views provided by ALTIBASE HDB:

```
iSQL> SELECT * FROM v$tab;
TABLE NAME                                TYPE
-----
V$ALLCOLUMN                              PERFORMANCE VIEW
V$ARCHIVE                                PERFORMANCE VIEW
V$BUFFPAGEINFO                           PERFORMANCE VIEW
V$BUFFPOOL_STAT                          PERFORMANCE VIEW
V$CATALOG                                PERFORMANCE VIEW
V$DATABASE                               PERFORMANCE VIEW
V$DATAFILES                              PERFORMANCE VIEW
V$DATATYPE                               PERFORMANCE VIEW
V$DBA_2PC_PENDING                        PERFORMANCE VIEW
V$DBLINK_REMOTE_STATEMENT_INFO           PERFORMANCE VIEW
V$DBLINK_REMOTE_TRANSACTION_INFO         PERFORMANCE VIEW
V$DBLINK_TRANSACTION_INFO                PERFORMANCE VIEW
V$DB_FREEPAGELISTS                       PERFORMANCE VIEW
V$DB_PROTOCOL                           PERFORMANCE VIEW
V$DIRECT_PATH_INSERT                     PERFORMANCE VIEW
V$DISKTBL_INFO                           PERFORMANCE VIEW
V$DISK_BTREE_HEADER                      PERFORMANCE VIEW
V$DISK_RTREE_HEADER                      PERFORMANCE VIEW
V$EVENT_NAME                             PERFORMANCE VIEW
V$FILESTAT                               PERFORMANCE VIEW
V$FLUSHER                                PERFORMANCE VIEW
V$FLUSHINFO                              PERFORMANCE VIEW
.
```

For the complete list of the performance views provided with ALTIBASE HDB and the meanings of the columns, please refer to the *ALTIBASE HDB General Reference Chapter 3: Data Dictionary*. Data in a particular performance view can be queried in the same way as an ordinary table using a SELECT statement, and using JOIN, etc., results can be output in various forms.

### 2.4.2 Viewing the List of Tables

Information on all of the tables that exist in the database can be retrieved using the following SELECT statement. The SYS\_TABLES\_ meta table is an internal system table that contains information about the database catalog provided by ALTIBASE HDB.

```
iSQL> SELECT * FROM system_.sys_tables_;
.
```

```
iSQL> SELECT * FROM tab;      -> This command is available in iSQL only.
USER NAME      TABLE NAME      TYPE
-----
```

## 2.4 Retrieving Information Related to the Database and Database Objects

### 2.4.3 Viewing a Table Structure

The following command is used to retrieve information on user-created tables:

```
DESC table_name;
```

```
CREATE TABLE departments (  
DNO          SMALLINT      PRIMARY KEY,  
DNAME        CHAR(30)      NOT NULL,  
DEP_LOCATION CHAR(9),  
MGR_NO       INTEGER );
```

iSQL> DESC departments; -> table\_name: The name of a table whose information (table structure) you want to know.

```
[ TABLESPACE : SYS_TBS_MEM_DATA ]  
[ ATTRIBUTE ]
```

NAME	TYPE		IS NULL
DNO	SMALLINT	FIXED	NOT NULL
DNAME	CHAR(30)	FIXED	NOT NULL
DEP_LOCATION	CHAR(9)	FIXED	
MGR_NO	INTEGER	FIXED	
[ INDEX ]			

NAME	TYPE	IS UNIQUE	COLUMN
_SYS_IDX_ID_122	BTREE	UNIQUE	DNO ASC
[ PRIMARY KEY ]			

DNO

### 2.4.4 Viewing Sequence Information

The following commands are used to obtain information on all sequences that exist in the database:

```
SELECT * FROM seq;
```

```
iSQL> CONNECT sys/manager;
```

Connect success.

```
iSQL> CREATE USER user1 IDENTIFIED BY user1;
```

Create success.

```
iSQL> CONNECT user1/user1;
```

Connect success.

```
iSQL> CREATE SEQUENCE seq1 MAXVALUE 100 CYCLE;
```

Create success.

```
iSQL> CREATE SEQUENCE seq2;
```

Create success.

```
iSQL> CONNECT sys/manager;
```

Connect success.

```
iSQL> CREATE SEQUENCE seq2 START WITH 20 INCREMENT BY 30;
```

Create success.

```
iSQL> CREATE SEQUENCE seq3 CACHE 40;
```

Create success.

```
iSQL> SELECT * FROM seq;      -> When accessing the database using the SYS account, information  
of all sequences will be displayed.
```

USER\_NAME

SEQUENCE_NAME		CURRENT_VALUE	
INCREMENT_BY	MIN_VALUE	MAX_VALUE	CYCLE

## 2.4 Retrieving Information Related to the Database and Database Objects

```

-----
CACHE_SIZE
-----
SYS
SEQ2
30          1          9223372036854775806      NO
20
SYS
SEQ3
1          1          9223372036854775806      NO
40
USER1
SEQ1
1          1          100                    YES
20
USER1
SEQ2
1          1          9223372036854775806      NO
20
4 rows selected.
iSQL> CONNECT user1/user1;
Connect success.
iSQL> SELECT * FROM seq;      -> Information of all sequences created by User 1 will be displayed.
SEQUENCE_NAME                CURRENT_VALUE
-----
INCREMENT_BY                MIN_VALUE                MAX_VALUE                CYCLE
-----
CACHE_SIZE
-----
SEQ1
1          1          100                    YES
20
SEQ2
1          1          9223372036854775806      NO
20
2 rows selected.

```

# 2.5 Controlling Transactions

## 2.5.1 Defining Transaction Modes

AUTOCOMMIT determines whether to automatically commit the results of a command at the time of execution.

```
iSQL> AUTOCOMMIT OFF; -> Commands are not automatically committed before being manually committed by the user.
```

```
Set autocommit off success.
```

```
iSQL> AUTOCOMMIT ON; -> Commands are automatically committed at the time of execution.
```

```
Set autocommit on success.
```

## 2.5.2 PLANCOMMIT

```
SET PLANCOMMIT [ON/OFF] ;
```

When EXPLAIN PLAN has been set to ON or ONLY, there is the possibility that the iSQL commands DESC; SELECT \* FROM TAB; or SELECT \* FROM SEQ; will be committed, even if AUTOCOMMIT has been set to OFF. This setting determines whether to commit them automatically.

This setting has been provided to overcome the misunderstanding where the user believes that such a command has not been prepared, but the system prepares the command in order to generate the execution plan. The command would then be committed, without the user knowing it, when a COMMIT command is executed later. When this value is OFF (which is the default) in a session for which EXPLAIN PLAN is ON (or ONLY) and AUTOCOMMIT is OFF, ALTIBASE HDB does not autocommit the above commands (DESC, SELECT \* FROM tab; or SELECT \* FROM seq;). When this value is ON, iSQL issues a special commit command to commit these commands.



## 2.6 File Management

### 2.6.1 Saving Results

iSQL enables results returned through iSQL to be saved in a designated file. In the following example, results are stored in the designated file, book.txt, using the SPOOL command.

To cancel this command, use the SPOOL OFF command.

```
iSQL> SPOOL book.txt
Spool start. [book.txt] -> All subsequently executed commands and their results will be written to
book.txt. The file is created in the current directory.
```

```
iSQL> SPOOL OFF
Spool Stop -> From this point on, no more commands or results will be saved in the file.
```

### 2.6.2 Running Scripts

#### 2.6.2.1 @ Command

```
@ file_name [.sql]
```

or

```
START file_name [.sql]
```

*file\_name[.sql]*: The script file to be executed. If the filename extension is omitted, iSQL assumes the default command file extension (.sql).

When this command is executed, iSQL executes all of the commands in the specified script file in sequence.

@command performs the same function as START.

- An EXIT or QUIT command in the script file terminates iSQL.
- The script file may include general SQL statements, iSQL commands, references to stored procedures, etc.

The following is an example in which the schema.sql script, which can be found in the \$ALTIBASE\_HOME/sample/APRE/schema directory, which is the current directory, is executed.

```
iSQL> START schema.sql -> The SQL statements in the file are executed.
```

or

```
iSQL> @schema.sql
```

When specifying a script file, you can use a question mark ("?") to indicate the ALTIBASE HDB home directory (\$ALTIBASE\_HOME) of the user account. The following is an example in which the schema.sql script, which can be found in the \$ALTIBASE\_HOME/sample/APRE/schema directory, is executed regardless of which directory is the current directory.

## 2.6 File Management

```
iSQL> @?/sample/APRE/schema/schema.sql
```

The question mark (“?”) can also be used with the following iSQL commands:

EDIT, SAVE, LOAD, SPOOL, START

The -- or /\* \*/ characters can be used to insert comments in script files. -- means that everything that follows until the end of the line will be handled as a comment, whereas comments that span several lines are placed between /\* and \*/.

### 2.6.2.2 @@ Command

```
@@ file_name [.sql]
```

*file\_name[.sql]*: This indicates the embedded script to be executed. If the extension is omitted, iSQL assumes the default command file extension(.sql).

Executes the specified script. The functionality of the @@ command is similar to that of the @ command.

This command searches for script files in the same path as the script currently being executed, and is thus useful for executing embedded scripts.

The @@ command can be used for the following purposes:

- If a script file that contains the text @@*file\_name.sql* is executed, iSQL looks for the file specified by *file\_name.sql*, and executes its contents in sequence.

*file\_name.sql* must be located in the same directory as the script file that called it. If no such file exists, iSQL raises an error.

- If a user inputs @@*file\_name.sql* at the iSQL prompt, the result will be the same as when using iSQL to execute *@file\_name.sql*.
- The script typically may include SQL statements, iSQL commands, or stored procedures.
- An EXIT or QUIT command in the script terminates iSQL.

The following is an example of the execution of a.sql, in which schema.sql is referenced, from the \$ALTIBASE\_HOME directory. In order for this example to be executed without error, a.sql must exist in the \$ALTIBASE\_HOME/sample/APRE/schema directory alongside schema.sql.

```
iSQL> @sample/APRE/schema/a.sql
```

```
$ cat a.sql  
@@schema.sql
```

*Note: The following chapter provides examples of editing the results of a query in an iSQL environment based on the tables created by execution of the above script (see appendix Schema).*

## 2.6.3 Saving SQL Statements

Of the commands currently in the iSQL buffer, the SAVE command saves the most recently executed one in a file.

This file will be created in the current directory.

```
iSQL> SELECT * FROM book;
iSQL> SAVE book.sql -> 'SELECT * FROM book;' is saved in the file book.sql.
Save completed.
```

## 2.6.4 Loading SQL Statements

This function loads the first command in the specified file to the last position in the iSQL buffer.

```
iSQL> LOAD book.sql
iSQL> SELECT * FROM book;
Load completed.
```

```
iSQL> / -> The results of execution of SELECT * FROM book; can be seen.
```

## 2.6.5 Saving DML Statements

Executed DML statements such as INSERT, UPDATE, DELETE and MOVE are saved in \$ALTIBASE\_HOME/trc/isql\_query.log.

Specify SET QUERYLOGGING ON to use this functionality and OFF to disable it.

```
iSQL> SET QUERYLOGGING ON; -> From this point on, all executed DML statements will be saved in
$ALTIBASE_HOME/trc/isql_query.log.
```

```
iSQL> CREATE TABLE t1 ( I1 INTEGER );
Create success.
iSQL> INSERT INTO t1 VALUES ( 1 );
1 row inserted.
iSQL> UPDATE t1 SET i1 = 2;
1 row updated.
iSQL> SELECT * FROM t1;
I1
-----
2
1 row selected.
iSQL> DELETE FROM t1;
1 row deleted.
iSQL> DROP TABLE t1;
Drop success.
iSQL> EXIT
```

```
$ cat $ALTIBASE_HOME/trc/isql_query.log -> All queries executed since SET QUERYLOGGING ON
was executed can be observed.
```

```
[2009/09/16 10:36:14] [127.0.0.1:20300 SYS] INSERT INTO t1 VALUES ( 1 )
[2009/09/16 10:36:25] [127.0.0.1:20300 SYS] UPDATE t1 SET i1 = 2
[2009/09/16 10:36:31] [127.0.0.1:20300 SYS] DELETE FROM t1
```

## 2.6.6 Editing Query Statements

### 2.6.6.1 Editing the Most Recent Query Statement

The command `ed` is provided for creating and editing files in iSQL.

If you execute `ed` without parameters, a temporary file named `iSQL.buf` containing the most recently executed query statements will be created, and the following screen will be visible. (To save

## 2.6 File Management

space, only a few of the blank lines that would be displayed on the screen are shown here.)

```
iSQL> SELECT sysdate FROM dual;
SYSDATE
-----
01-JAN-2000
1 row selected.

iSQL> ED
SELECT sysdate FROM dual;
~
~
~
"iSQL.buf" 1L, 26C
```

### 2.6.6.2 Editing Existing Files

If you want to edit an existing file, type the file name in iSQL as a parameter when launching the text editor using the “ed” command. When the screen is initialized, blank lines will be displayed as ~ (tilde) characters.

```
iSQL> ED myquery.sql
"myquery.sql"
INSERT INTO employees(ENO, E_FIRSTNAME, E_LASTNAME, SEX) VALUES(21, 'Shiloh',
'Reynolds', 'F');
INSERT INTO employees(ENO, E_FIRSTNAME, E_LASTNAME, SEX, JOIN_DATE) VAL-
UES(22, 'Joshua', 'Baldwin', 'M', TO_DATE('2001-11-19 00:00:00', 'YYYY-MM-DD
HH:MI:SS'));
~
~"myquery.sql"
```

### 2.6.6.3 Editing Query Statements in History Lists

You can use the number in the history list to edit previously executed commands. In detail, the query statements are stored in the temporary file iSQL.buf in association with numbers, and can be edited with reference to them. The edited query will be stored again as the most recent record in the history list, and can be executed by entering the ‘/’ (re-execute) character.

```
iSQL> H
1 : SELECT * FROM customers;
2 : SELECT * FROM employees;

iSQL> 2ed

or

iSQL> 2 ed
SELECT * FROM employees;
~
~
"iSQL.buf"
```

*Note: The command-line parameter 2, which is the name of the file to be edited (iSQL> ed 2), must be distinguished from the number indicating the line in the file to edit.*

After editing (*employees* was replaced with *orders*)

```
iSQL> h -> The history list currently in the isql buffer
1 : SELECT * FROM customers;
2 : SELECT * FROM employees;
```

3 : SELECT \* FROM orders; -> The query statement edited using the 2 ed command will be saved as the last command in the history list.

iSQL> / -> The most recently executed command will be executed.

ONO		ORDER_DATE	ENO	CNO
-----				
GNO	QTY	ARRIVAL_DATE	PROCESSING	
-----				
11290007		29-NOV-2010	12	7111111431202
A111100002	70	02-DEC-2010	C	
11290011		29-NOV-2010	12	7610011000001
E111100001	1000	05-DEC-2010	D	
11290100		29-NOV-2010	19	7001011001001
E111100001	500	07-DEC-2010	D	
12100277		10-DEC-2010	19	7610121220475
.				
.				
12310012		31-DEC-2010	19	7308281201145
C111100001	250	03-JAN-2011	O	
30 rows selected.				

#### 2.6.6.4 Environment Variables for the Default Editor

You can set the environment variable ISQL\_EDITOR to use an editor other than the default editor (/usr/bin/vi).

Ex)

CSH: setenv ISQL\_EDITOR usr/bin/ed

SH: ISQL\_EDITOR=/usr/bin/ed; export ISQL\_EDITOR

## 2.7 Formatting SELECT Query Results

The results of a SELECT query can be formatted as desired by the user.

### 2.7.1 SET LINESIZE

Sets the size (number of characters) of one line to be displayed when the results of a SELECT statement are output. It must be between 10 and 32767.

```
iSQL> set linesize 70;
iSQL> select * from employees;
ENO          E_LASTNAME          E_FIRSTNAME
-----
EMP_JOB      EMP_TEL      DNO      SALARY      SEX
-----
BIRTH      JOIN_DATE      STATUS
-----
1          Moon          Chan-seung
CEO          01195662365      3002          M
          R
2          Davenport      Susan
designer      0113654540      1500          F
721219  18-NOV-2009  H
.
.
20 rows selected.
```

### 2.7.2 SET LOBSIZE

This specifies the number of characters to display when a CLOB column is queried using a SELECT statement.

In order to query CLOB column data using a SELECT statement, the transaction mode must first be set to AUTOCOMMIT OFF.

```
iSQL> CREATE TABLE c1(I1 INTEGER, I2 CLOB);
INSERT INTO c1 VALUES(1, 'A123456789');
INSERT INTO c1 VALUES(2, 'A1234');
INSERT INTO c1 VALUES(3, 'A12345');
INSERT INTO c1 VALUES(4, 'A1234567890123');

iSQL> AUTOCOMMIT OFF -> This sets the transaction mode to OFF so that a CLOB column can be queried.
Set autocommit off success.

iSQL> SELECT * FROM c1;
I1          I2
-----
1          A123456789
2          A1234
3          A12345
4          A1234567890123

4 rows selected.
```

iSQL> SET LOBSIZE 10; -> This specifies the number of characters to display on the screen when querying a CLOB column using a SELECT statement.

```
iSQL> SELECT * FROM c1;
I1          I2
-----
1          A123456789
2          A1234
3          A12345
4          A123456789
4 rows selected.
```

### 2.7.3 SET LOBOFFSET

This specifies the starting location from which to display CLOB data when a CLOB column is queried using a SELECT statement.

In order to query CLOB column data using a SELECT statement, the transaction mode must first be set to AUTOCOMMIT OFF.

```
iSQL> CREATE TABLE c1(I1 INTEGER, I2 CLOB);
INSERT INTO c1 VALUES(1, 'A123456789');
INSERT INTO c1 VALUES(2, 'A1234');
INSERT INTO c1 VALUES(3, 'A12345');
INSERT INTO c1 VALUES(4, 'A1234567890123');
```

```
iSQL> AUTOCOMMIT OFF
Set autocommit off success.
```

iSQL> SET LOBOFFSET 4; -> This specifies the starting location of data to be shown on the screen number of characters to skip) when querying a CLOB column using a SELECT statement.

```
iSQL> SELECT * FROM c1;
I1          I2
-----
1          456789
2          4
3          45
4          4567890123
4 rows selected.
```

### 2.7.4 SET FEEDBACK

Outputs the number of records found when the results of a SELECT statement are output.

```
SET FEEDBACK ON|OFF|n;
```

ON: Output the number of resultant records after execution of a SELECT statement.

OFF: Do not output the number of resultant records after execution of a SELECT statement.

n: Output the number of resultant records when the number is n or greater.

```
iSQL> set feedback on;
iSQL> select * from employees where eno < 3;
ENO          E_LASTNAME          E_FIRSTNAME
-----
EMP_JOB      EMP_TEL      DNO          SALARY      SEX
-----
BIRTH      JOIN_DATE      STATUS
```

## 2.7 Formatting SELECT Query Results

```
-----
1          Moon          Chan-seung
CEO          01195662365  3002          M
              R
2          Davenport     Susan
designer      0113654540          1500          F
721219 18-NOV-2009 H
2 rows selected.
```

### 2.7.5 SET PAGESIZE

Specifies the number of resultant rows to display at one time.

```
iSQL> SET PAGESIZE 2; -> Show results in groups comprising two rows each.
```

```
iSQL> select eno, e_firstname, e_lastname from employees;
```

```
ENO          E_FIRSTNAME          E_LASTNAME
-----
1          Chan-seung          Moon
2          Susan          Davenport
ENO          E_FIRSTNAME          E_LASTNAME
-----
3          Ken          Kobain
4          Aaron          Foster
ENO          E_FIRSTNAME          E_LASTNAME
-----
5          Farhad          Ghorbani
6          Ryu          Momoi
.
.
.
20 rows selected.
```

```
iSQL> SET PAGESIZE 0; -> Show all of the results on one page.
```

```
iSQL> select eno, e_firstname, e_lastname from employees;
```

```
ENO          E_FIRSTNAME          E_LASTNAME
-----
1          Chan-seung          Moon
2          Susan          Davenport
3          Ken          Kobain
4          Aaron          Foster
5          Farhad          Ghorbani
6          Ryu          Momoi
.
.
.
20 rows selected.
```

### 2.7.6 SET HEADING

Sets whether to output the header with a SELECT result.

```
iSQL> SET HEADING OFF; -> Header is not displayed with the result.
```

```
iSQL> select eno, e_firstname, e_lastname from employees;
```

```
1          Chan-seung          Moon
2          Susan          Davenport
3          Ken          Kobain
4          Aaron          Foster
5          Farhad          Ghorbani
6          Ryu          Momoi
```



```

.
.
.
20 rows selected.

iSQL> SET HEADING ON; -> Outputs header in result.
iSQL> select eno, e_firstname, e_lastname from employees;
ENO          E_FIRSTNAME          E_LASTNAME
-----
1            Chan-seung            Moon
2            Susan              Davenport
3            Ken                Kobain
4            Aaron              Foster
5            Farhad             Ghorbani
6            Ryu               Momoi
.
.
.
20 rows selected.

```

### 2.7.7 SET COLSIZE

When the results of a SELECT statement are output, sets the number of characters from a column of type CHAR or VARCHAR to display so that columns containing long lines of text can be easily viewed.

```

iSQL> CREATE TABLE location(
ID          INTEGER,
NAME        CHAR(20),
ADDRESS     VARCHAR(500),
PHONE       CHAR(20));
Create success.
iSQL> INSERT INTO location VALUES(1, 'ALTIBASE', '10Fl., Daerungpost-tower
II, Guro-dong, Guro-qu, Seoul 152-790. Korea', '82-2-2082-1000');
1 row inserted.

```

In the following example, the number of characters of a column of type CHAR or VARCHAR is set to 7:

```

iSQL> SET COLSIZE 7;
iSQL> SELECT id, name, address, phone FROM location;
ID          NAME          ADDRESS          PHONE
-----
1            ALTIBAS  10Fl.,      82-2-20
            E          Daerung  82-1000
            post-to
            wer II,
            Guro-d
            ong, Gu
            ro-qu,
            Seoul 1
            52-790.
            Korea
1 row selected.

```

## 2.8 Setting Output Options

### 2.8.1 Getting the Elapsed Time

This function displays the time it took to execute the SQL statement.

```
iSQL> SET TIMING ON; -> Output the execution time in the last line after the command is executed.
iSQL> select eno, e_firstname, e_lastname from employees;
ENO          E_FIRSTNAME      E_LASTNAME
-----
1            Chan-seung        Moon
2            Susan            Davenport
3            Ken              Kobain
4            Aaron            Foster
5            Farhad          Ghorbani
6            Ryu              Momoi
.
.
.
20 rows selected.
elapsed time : 0.01
iSQL> SET TIMING OFF; -> Execution time is not displayed.
```

### 2.8.2 Setting Execution Time Units for Output

This function sets the units with which to output SQL statement execution time. Can be set to the following units:

- Seconds
- Milliseconds
- Microseconds
- Nanoseconds

```
iSQL> SET TIMING ON
iSQL> SET TIMESCALE SEC;
iSQL> select eno, e_firstname, e_lastname from employees;
ENO          E_FIRSTNAME      E_LASTNAME
-----
1            Chan-seung        Moon
2            Susan            Davenport
3            Ken              Kobain
4            Aaron            Foster
5            Farhad          Ghorbani
6            Ryu              Momoi
.
.
.
20 rows selected.
elapsed time : 0.00
```

```
iSQL> SET TIMESCALE MILSEC;
iSQL> select eno, e_firstname, e_lastname from employees;
ENO          E_FIRSTNAME      E_LASTNAME
-----
1            Chan-seung        Moon
```

```

2          Susan          Davenport
3          Ken            Kobain
4          Aaron          Foster
5          Farhad         Ghorbani
6          Ryu            Momoi
.
.
.

```

```

20 rows selected.
elapsed time : 0.72

```

```

iSQL> SET TIMESCALE MICSEC;
iSQL> select eno, e_firstname, e_lastname from employees;

```

```

ENO          E_FIRSTNAME          E_LASTNAME
-----
1          Chan-seung          Moon
2          Susan            Davenport
3          Ken              Kobain
4          Aaron            Foster
5          Farhad           Ghorbani
6          Ryu              Momoi
.
.
.

```

```

20 rows selected.
elapsed time : 966.00

```

```

iSQL> SET TIMESCALE NANSEC;
iSQL> select eno, e_firstname, e_lastname from employees;

```

```

ENO          E_FIRSTNAME          E_LASTNAME
-----
1          Chan-seung          Moon
2          Susan            Davenport
3          Ken              Kobain
4          Aaron            Foster
5          Farhad           Ghorbani
6          Ryu              Momoi
.
.
.

```

```

20 rows selected.
elapsed time : 681000.00

```

### 2.8.3 Describing Foreign Key Information

This function displays information on foreign keys when the DESC command is used to view the table structure.

```

iSQL> SET FOREIGNKEYS ON; -> The foreign key information will be output.

```

```

iSQL> DESC bikes_ive_seen;
[ TABLESPACE : SYS_TBS_MEM_DATA ]
[ ATTRIBUTE ]

```

```

-----
NAME                                TYPE                                IS NULL
-----
MID                                SMALLINT                            FIXED
YEAR                                SMALLINT                            FIXED      NOT NULL
USED                                BIT(1)                             FIXED      NOT NULL
SOLD                                BIT(1)                             FIXED
KMS                                INTEGER                            FIXED
SAW WHERE                          VARCHAR(20)                         FIXED
ITEM_ID                            INTEGER                            FIXED      NOT NULL

```

## 2.8 Setting Output Options

```
COMMENT                                VARCHAR(100)    FIXED
PRICE                                INTEGER         FIXED      NOT NULL
DATE_SEEN                            DATE            FIXED
[ INDEX ]

-----
NAME                                TYPE           IS UNIQUE    COLUMN
-----
__SYS_IDX_ID_143                    BTREE          UNIQUE       ITEM_ID ASC
[ PRIMARY KEY ]

-----
ITEM_ID

[ FOREIGN KEYS ]

-----
* MODEL_ID                          * __SYS_IDX_ID_142
( MID )                            ---> SYS.CANDIDATE_MODELS ( MID )
iSQL> SET FOREIGNKEYS OFF; -> The foreign key information will not be output.
iSQL> DESC bikes_ive_seen;
[ TABLESPACE : SYS_TBS_MEM_DATA ]
[ ATTRIBUTE ]

-----
NAME                                TYPE           IS NULL
-----
MID                                SMALLINT       FIXED
YEAR                                SMALLINT       FIXED      NOT NULL
USED                                BIT(1)         FIXED      NOT NULL
SOLD                                BIT(1)         FIXED
KMS                                INTEGER        FIXED
SAW_WHERE                          VARCHAR(20)    FIXED
ITEM_ID                            INTEGER        FIXED      NOT NULL
COMMENT                            VARCHAR(100)   FIXED
PRICE                                INTEGER        FIXED      NOT NULL
DATE_SEEN                            DATE           FIXED
[ INDEX ]

-----
NAME                                TYPE           IS UNIQUE    COLUMN
-----
__SYS_IDX_ID_143                    BTREE          UNIQUE       ITEM_ID ASC
[ PRIMARY KEY ]

-----
ITEM_ID
iSQL>
```

### 2.8.4 Outputting Script Execution Results

Commands can be used to control the output of created results.

When set to OFF, this function prevents the result of a script command executed in iSQL from being displayed on the screen.

However, even if it is set OFF, the results of queries that are directly entered (e.g. iSQL> select \* from t1;) will still be displayed on the screen; the OFF setting only prevents script execution results (e.g.: iSQL> @a.sql ) from being displayed.

```
iSQL> SET TERM OFF;
iSQL> SET TIMING ON; -> The execution time is not output to the screen.
iSQL> @schema.sql -> The script execution results are not output.
iSQL> select eno, e_firstname, e_lastname from employees;
-> The results of directly input queries will be output.
ENO          E_FIRSTNAME          E_LASTNAME
-----
```

```

1          Chan-seung          Moon
2          Susan               Davenport
3          Ken                 Kobain
4          Aaron               Foster
5          Farhad              Ghorbani
6          Ryu                 Momoi
.
.
.
20 rows selected.
elapsed time : 0.00

iSQL> SET TERM ON; -> Script execution results will be output.
iSQL> @schema.sql
iSQL> ALTER SESSION SET AUTOCOMMIT = TRUE; -> Start of results.
Alter success.
iSQL> DROP TABLE ORDERS;
Drop success.
elapsed time : 0.00
iSQL> DROP TABLE EMPLOYEES;
Drop success.
elapsed time : 0.00
.
.
.
iSQL> CREATE INDEX ODR_IDX3 ON ORDERS (GNO ASC);
Create success.
elapsed time : 0.00 -> End of results.

```

### 2.8.5 Outputting an Execution Plan

In iSQL, an execution plan can be output to fine-tune SQL statements. Using an execution plan, DML statements such as SELECT, INSERT, UPDATE and DELETE can be checked.

In order to accomplish this, the following command must be executed before a statement such as a SELECT statement is executed.

```
ALTER SESSION SET EXPLAIN PLAN = option
```

This option can be set to ON, OFF or ONLY. The default is OFF.

- **ON:** After the SELECT statement is executed, the execution plan information is displayed along with the resultant records.
- **ONLY:** The SELECT statement is prepared but not executed, and only the execution plan information is output. This can be used to check the execution plan for a SELECT statement that involves host variable binding, or to quickly check the execution plan for queries that take a long time to execute.
- **OFF:** After the SELECT statement is executed, only the resultant records are displayed.

The following command is used to obtain detailed information about how conditions included in WHERE clauses written by the user will be executed:

```
ALTER SYSTEM SET TRCLOG_DETAIL_PREDICATE = 1
```

If this property is set to 1, signifying "ON", as in the above statement, the execution plan's WHERE clause conditions, including FIXED KEY RANGE, VARIABLE KEY RANGE and FILTER, are classified and displayed in detail. Therefore, if the WHERE clause is complicated, you can check which predicates

## 2.8 Setting Output Options

will be executed by scanning the sorted indexes. However, this information may not be output if queries are changed to optimize them in some way.

The following example shows the output when the given SQL statement is executed:

- When TRCLOG\_DETAIL\_PREDICATE has been set to 1 (=on), and EXPLAIN PLAN = ON, the following is output in addition to the results.

```
iSQL> ALTER SYSTEM SET TRCLOG_DETAIL_PREDICATE = 1;
Alter success.
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
Alter success.
iSQL> select eno, e_lastname, e_firstname from employees where eno = 1;
ENO          E_LASTNAME          E_FIRSTNAME
-----
1            Moon            Chan-seung
1 row selected.
-----
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 48 )
  SCAN ( TABLE: EMPLOYEES, INDEX: __SYS_IDX_ID_164, ACCESS: 1, SELF_ID: 2
)
  [ FIXED KEY ]
  AND
  OR
  ENO = 1
-----

iSQL>
```

- When TRCLOG\_DETAIL\_PREDICATE is not set to 1, and EXPLAIN PLAN = ON, the following is output in addition to the results.

```
iSQL> ALTER SYSTEM SET TRCLOG_DETAIL_PREDICATE = 0;
Alter success.
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
Alter success.
iSQL> select eno, e_lastname, e_firstname from employees where eno = 1;
ENO          E_LASTNAME          E_FIRSTNAME
-----
1            Moon            Chan-seung
1 row selected.
-----
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 48 )
  SCAN ( TABLE: EMPLOYEES, INDEX: __SYS_IDX_ID_164, ACCESS: 1, SELF_ID: 2
)
-----

iSQL>
```

- When TRCLOG\_DETAIL\_PREDICATE is not set to 1, and EXPLAIN PLAN = ONLY, only the following is output.

```
iSQL> ALTER SYSTEM SET TRCLOG_DETAIL_PREDICATE = 0;
Alter success.
iSQL> ALTER SESSION SET EXPLAIN PLAN = ONLY;
Alter success.
iSQL> select eno, e_lastname, e_firstname from employees where eno = 1;
ENO          E_LASTNAME          E_FIRSTNAME
-----
No rows selected.
-----
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 48 )
  SCAN ( TABLE: EMPLOYEES, INDEX: __SYS_IDX_ID_164, ACCESS: 1, SELF_ID: 2
)
-----
```

```
iSQL>
```

If EXPLAIN PLAN = ONLY, because only an execution plan is created and the query is not executed, values that would be determined after actual execution are indicated using question marks ("??"), like an ACCESS clause.

### 2.8.6 Setting Result Output Orientation

When querying data using a SELECT statement in iSQL, the results can be displayed either horizontally or vertically.

This function is suitable for outputting results that comprise a small number of rows and many columns.

If such a result set is output horizontally, as is usually the case, it is difficult to compare columns and check the values. However, it is easy to see when output vertically.

```
iSQL> SET VERTICAL ON; --> This sets the print direction vertically.
```

```
iSQL> SELECT * FROM employees WHERE eno = 2;
```

```
ENO          : 2
E_LASTNAME   : Davenport
E_FIRSTNAME  : Susan
EMP_JOB      : designer
EMP_TEL      : 0113654540
DNO          :
SALARY       : 1500
SEX          : F
BIRTH        : 721219
JOIN_DATE    : 18-NOV-2009
STATUS       : H
```

```
1 row selected.
```

# 2.9 Viewing iSQL Display Settings

The following is an example of viewing the values of the iSQL environment variables for the current session:

```
iSQL> SHOW USER -> This is the current user.
```

```
User : SYS
```

```
iSQL> SHOW COLSIZE
```

```
ColSize : 0
```

```
iSQL> SHOW LOBOFFSET
```

```
LobOffset: 0
```

```
iSQL> SHOW LINESIZE
```

```
LineSize : 80
```

```
iSQL> SHOW LOBSIZE
```

```
LobSize : 80
```

```
iSQL> SHOW PAGESIZE
```

```
PageSize : 0
```

```
iSQL> SHOW TIMESCALE
```

```
TimeScale : Second
```

```
iSQL> SHOW HEADING
```

```
Heading : On
```

```
iSQL> SHOW TIMING
```

```
Timing : Off
```

```
iSQL> SHOW VERTICAL
```

```
Vertical : Off
```

```
iSQL> SHOW FOREIGNKEYS
```

```
ForeignKeys : Off
```

```
iSQL> SHOW PLANCOMMIT
```

```
PlanCommit : Off
```

```
iSQL> SHOW QUERYLOGGING
```

```
QueryLogging : Off
```

```
iSQL> SHOW TERM
```

```
Term : On
```

```
iSQL> SHOW FEEDBACK
```

```
Feedback : 1
```

```
iSQL> SHOW ALL
```

```
User : SYS
```

```
ColSize : 0
```

```
LobOffset : 0
```

```
LineSize : 80
```

```
LobSize : 80
```

```
PageSize : 0
```

```
TimeScale : Second
```

```
Heading : On
```

```
Timing : Off
```

```
Vertical : Off
```

```
ForeignKeys : Off
```

```
PlanCommit : Off
```



```
QueryLogging : Off  
Term : On  
Feedback : 1
```

# 2.10 Host Variables

Host variables are first declared and then used. Host variables are useful when executing procedures or functions.

## 2.10.1 Declaring a Host Variable

### 2.10.1.1 Syntax

```
VAR[iable] var_name var_type
```

### 2.10.1.2 Type

The following types can be used when declaring variables:

```
INTEGER, BYTE(n), NIBBLE(n),  
NUMBER, NUMBER(n), NUMBER(n,m),  
NUMERIC, NUMERIC(n), NUMERIC(n,m),  
CHAR(n), VARCHAR(n), NCHAR(n), NVARCHAR(n), DATE  
DECIMAL, DECIMAL(n), DECIMAL(n,m),  
FLOAT, FLOAT(n), DOUBLE, REAL  
BIGINT, SMALLINT
```

### 2.10.1.3 Example

The following examples demonstrate how to declare variables:

```
iSQL> VAR p1 INTEGER  
iSQL> VAR p2 CHAR(10)  
iSQL> VAR v_double DOUBLE  
iSQL> VAR v_real REAL
```

## 2.10.2 Assigning a Value to a Host Variable

### 2.10.2.1 Syntax

```
EXEC[UTE] :var_name := value;
```

### 2.10.2.2 Example

The following example shows how to assign a value to a variable:

```
iSQL> EXECUTE :p1 := 100;  
Execute success.  
iSQL> EXEC :p2 := 'abc';  
Execute success.
```

## 2.10.3 Viewing Host Variables

### 2.10.3.1 Syntax

```
PRINT VAR [IABLE]
```

-> Shows all declared variables.

```
PRINT var_name
```

-> Shows the type and value of the variable *var\_name*.

### 2.10.3.2 Example

The following shows the values of all declared variable:

```
iSQL> PRINT VAR  
[ HOST VARIABLE ]
```

NAME	TYPE	VALUE
P1	INTEGER	100
P2	CHAR (10)	abc
V_REAL	REAL	
V_DOUBLE	DOUBLE	

```
iSQL> PRINT p2 -> Outputs only variable p2 information.
```

NAME	TYPE	VALUE
P2	CHAR (10)	abc

# 2.11 Executing Prepared SQL Statements

### 2.11.1 Prepared SQL versus Dynamic SQL Statements

SQL statements executed in iSQL are usually executed according to the so-called “Direct Execution” method.

In Direct Execution, syntax analysis, validity testing, optimization and execution of a query are all performed at once. However, in Prepared Execution, only the syntax analysis, validity testing, and optimization of the query are performed to set up an execution plan for the query, which is then executed when requested by the client. When creating an application that uses ODBC, the Prepared Execution method is typically used, and is more advantageous in terms of speed when an SQL statement is to be repeatedly executed using host variable binding.

In iSQL, the difference between these two methods lies only in whether variables are used or not; there is no advantage in terms of speed.

### 2.11.2 Prepared SQL Statements

#### 2.11.2.1 Syntax

```
PREPARE SQL_statement
```

#### 2.11.2.2 Example

The following is an example of the use of the PREPARE command to execute an SQL statement:

```
iSQL> VAR t1 INTEGER;
iSQL> EXEC :t1 := 3;
Execute success.
iSQL> PREPARE SELECT eno, e_firstname, e_lastname, sex
FROM employees
WHERE eno=:t1;
ENO
: 3
E_FIRSTNAME : Ken
E_LASTNAME  : Kobain
SEX         : M

1 row selected.
```

## 2.12 Creating, Executing and Dropping Stored Procedures

### 2.12.1 Creating Procedures

Support is provided for the creation and execution of stored procedures. A stored procedure must end with the following:

```
END;
/
```

Successful creation of the procedures can be confirmed by checking the `sys_procedures_meta` table.

### 2.12.2 Executing Procedures

Procedures are executed in order to execute multiple queries at one time. If the procedure to be executed has parameters, as many variables as there are parameters must be declared before the procedure is executed.

#### 2.12.2.1 Example 1

In the following example, a procedure named `emp_proc`, which executes an `INSERT` statement using `IN` parameters, is created:

```
iSQL> CREATE OR REPLACE PROCEDURE emp_proc(p1 IN INTEGER, p2 IN CHAR(20), p3
IN CHAR(20), p4 IN CHAR(1))
AS
BEGIN
  INSERT INTO employees(eno, e_firstname, e_lastname, sex)
  VALUES(p1, p2, p3, p4);
END;
/
Create success.
```

```
iSQL> SELECT * FROM system.sys_procedures_ order by created desc limit 1;
USER_ID      PROC_OID
```

```
-----
PROC_NAME                                OBJECT_TYPE STATUS
-----
-----
PARA_NUM      RETURN_DATA_TYPE RETURN_LANG_ID RETURN_SIZE
-----
RETURN_PRECISION RETURN_SCALE PARSE_NO      PARSE_LEN   CREATED
-----
LAST_DDL_TIME
-----
2              3208680
EMP_PROC                                0           0
4
              2              192          29-FEB-2012
29-FEB-2012
1 row selected.
```

*emp\_proc*, which was created above, is executed:

## 2.12 Creating, Executing and Dropping Stored Procedures

```
iSQL> VAR eno INTEGER
iSQL> VAR first_name CHAR(20)
iSQL> VAR last_name CHAR(20)
iSQL> VAR sex CHAR(1)
iSQL> EXECUTE :eno := 21;
Execute success.
iSQL> EXECUTE :first_name := 'Joel';
Execute success.
iSQL> EXECUTE :last_name := 'Johnson';
Execute success.
iSQL> EXECUTE :sex := 'M';
Execute success.
iSQL> EXECUTE emp_proc(:eno, :firstname, :lastname, :sex);
Execute success.
iSQL> SELECT eno, e_firstname, e_lastname, sex FROM employees WHERE eno = 21;
ENO          E_FIRSTNAME          E_LASTNAME          SEX
-----
21          Joel          Johnson          M
1 row selected.
```

### 2.12.2.2 Example 2

In the following example, a procedure called outProc, which executes a SELECT statement, is created:

```
iSQL> CREATE TABLE outTbl(i1 INTEGER, i2 INTEGER);
Create success.
iSQL> INSERT INTO outTbl VALUES(1,1);
1 row inserted.
iSQL> /
1 row inserted.
iSQL> /
1 row inserted.
iSQL> /
1 row inserted.
iSQL> /
1 row inserted.
iSQL> SELECT * FROM outTbl;
I1          I2
-----
1          1
1          1
1          1
1          1
1          1
5 rows selected.
iSQL> CREATE OR REPLACE PROCEDURE outProc(a1 OUT INTEGER, a2 IN OUT INTEGER)
AS
BEGIN
    SELECT COUNT(*) INTO a1 FROM outTbl WHERE i2 = a2;
END;
/
Create success.
```

In the following example, outProc is executed:

```
iSQL> VAR t3 INTEGER
iSQL> VAR t4 INTEGER
iSQL> EXEC :t4 := 1;
Execute success.
iSQL> EXEC outProc (:t3, :t4);
Execute success.
iSQL> PRINT t3;
```

NAME	TYPE	VALUE
T3	INTEGER	5

### 2.12.2.3 Example 3

In the following example, the procedure outProc1 is created:

```
iSQL> CREATE OR REPLACE PROCEDURE outProc1( p1 INTEGER, p2 IN OUT INTEGER, p3
OUT INTEGER)
AS
BEGIN
  p2 := p1;
  p3 := p1 + 100;
END;
/
Create success.
```

```
iSQL> VAR v1 INTEGER
iSQL> VAR v2 INTEGER
iSQL> VAR v3 INTEGER
iSQL> EXEC :v1 := 3;
Execute success.
iSQL> EXEC outProc1(:v1, :v2, :v3);
Execute success.
```

```
iSQL> PRINT VAR;
[ HOST VARIABLE ]
```

NAME	TYPE	VALUE
.		
.		
V1	INTEGER	3
V2	INTEGER	3
V3	INTEGER	103
.		
.		

### 2.12.2.4 Example 4

In the following example, a procedure called inoutProc1, which executes a SELECT statement, is created:

```
iSQL> CREATE TABLE inoutTbl(i1 INTEGER);
Create success.
iSQL> INSERT INTO inoutTbl VALUES(1);
1 row inserted.
iSQL> /
1 row inserted.
iSQL> /
1 row inserted.
iSQL> SELECT * FROM inoutTbl;
I1
-----
1
1
1
3 rows selected.
iSQL> CREATE OR REPLACE PROCEDURE inoutProc (a1 IN OUT INTEGER)
AS
```

## 2.12 Creating, Executing and Dropping Stored Procedures

```
BEGIN
  SELECT COUNT(*) INTO a1 FROM inoutTbl WHERE i1 = a1;
END;
/
Create success.
```

```
iSQL> VAR t3 INTEGER
iSQL> EXEC :t3 := 1;
Execute success.
iSQL> EXEC inoutProc(:t3);
Execute success.
```

```
iSQL> PRINT t3;
NAME                TYPE                VALUE
-----
T3                   INTEGER              3
```

### 2.12.2.5 Example 5

In the following example, the procedure inoutProc1 is created:

```
iSQL> CREATE OR REPLACE PROCEDURE inoutProc1( p1 INTEGER, p2 IN OUT INTEGER,
p3 OUT INTEGER)
AS
BEGIN
  p2 := p1 + p2;
  p3 := p1 + 100;
END;
/
Create success.
```

In the following example, the procedure inoutProc1 is executed:

```
iSQL> VAR v1 INTEGER
iSQL> VAR v2 INTEGER
iSQL> VAR v3 INTEGER
iSQL> EXEC :v1 := 3;
Execute success.

iSQL> EXEC :v2 := 5;
Execute success.

iSQL> EXEC inoutProc1(:v1, :v2, :v3);
Execute success.
```

```
iSQL> PRINT VAR;
[ HOST VARIABLE ]
-----
NAME                TYPE                VALUE
-----
.
.
V1                   INTEGER              3
V2                   INTEGER              8
V3                   INTEGER             103
.
.
```

### 2.12.3 Dropping Procedures

The DROP command is used to drop (delete) procedures.



## 2.12 Creating, Executing and Dropping Stored Procedures

In the following example, the procedure emp\_proc is deleted:

```
iSQL> DROP PROCEDURE emp_proc;  
Drop success.
```

## 2.13 Creating, Executing and Dropping Functions

### 2.13.1 Creating Functions

A function is provided to create functions. When creating a function, you must end with the following syntax, and the return type must be defined.

```
END;
/
```

Successful creation of the function can be confirmed by checking the `sys_procedures_meta` table.

In the following example, the function `emp_func`, which executes an `UPDATE` statement and a `SELECT` statement, is created:

```
iSQL> CREATE OR REPLACE FUNCTION emp_func(f1 IN INTEGER)
RETURN NUMBER
AS
  f2 NUMBER;
BEGIN
  UPDATE employees SET salary = 1000000 WHERE eno = f1;
  SELECT salary INTO f2 FROM employees WHERE eno = f1;
  RETURN f2;
END;
/
Create success.
```

```
iSQL> SELECT * FROM system.sys_procedures_;
USER_ID      PROC_OID      PROC_NAME
-----
---
OBJECT_TYPE  STATUS        PARA_NUM      RETURN_DATA_TYPE RETURN_LANG_ID
-----
RETURN_SIZE  RETURN_PRECISION RETURN_SCALE  PARSE_NO        PARSE_LEN
-----
CREATED      LAST_DDL_TIME
-----
.
.
.
2            3300024              INOUTPROC1
0            0                    3
                2                    132
15-SEP-2010  15-SEP-2010
2            3302344              EMP_FUNC
1            0                    1                    6                    30000
23           38                    0                    3                    209
15-SEP-2010  15-SEP-2010
36 rows selected.
```

### 2.13.2 Executing Functions

Functions can be executed to simultaneously execute multiple queries. If the function to be executed has parameters, as many variables as there are functions must be declared before the function is executed. Additionally, a variable for saving the result of the function must also be defined.

The following is an example of executing the function emp\_func:

```
iSQL> VAR eno INTEGER
iSQL> VAR ret NUMBER
iSQL> EXEC :eno := 11;
Execute success.
iSQL> EXEC :ret := emp_func(:eno);
Execute success.

iSQL> SELECT eno, salary FROM employees WHERE eno = 11;
ENO          SALARY
-----
11           1000000
1 row selected.
```

### 2.13.3 Dropping Functions

The DROP FUNCTION statement is used to drop functions.

In the following example, the function emp\_func is deleted:

```
iSQL> DROP FUNCTION emp_func;
Drop success.
```

## 2.14 Convenient User Functions

### 2.14.1 History

A list of all previously executed commands can be displayed using the HISTORY command. The number corresponding to a previously executed command can be used to easily execute that command again.

```
iSQL> HISTORY; ->View history list.
```

or

```
iSQL> H;
1 : SELECT * FROM tab;
2 : SELECT * FROM v$tab;
```

```
iSQL> / -> Re-execute the most recent command(HISTORY;)
```

```
iSQL> 2/ -> Execute Command number 2 in history list(SELECT * FROM book;)
```

### 2.14.2 Shell Commands

The exclamation point ("!") is a convenient function that allows direct execution of most shell commands from within iSQL.

```
iSQL> !ls -al
total 3417
-rw-r----- 1 wlgml337 section 1198 Nov 1 13:30 .aliases
-rw----- 1 wlgml337 section 5353 Oct 18 21:17 .bash_history
-rw-r----- 1 wlgml337 section 1436 Nov 2 15:42 .bashrc
-rw-r----- 1 wlgml337 section 1549 Dec 13 17:36 .profile
drwxr-x--- 2 wlgml337 section 512 Nov 2 02:00 TEMP
drwxr-xr-x 2 root root 512 Oct 16 11:29 TT_DB
-rw----- 1 wlgml337 section 3446548 Dec 18 13:19 core
drwxr-x--- 2 wlgml337 section 512 Nov 11 16:33 cron
drwxr-x--- 2 wlgml337 section 512 Nov 15 10:52 test
drwxr-xr-x 6 wlgml337 section 512 Nov 11 11:45 work
```

### 2.14.3 Getting Help

Help is available for the commands provided with iSQL. The HELP command without parameters outputs information on how to use help. For help on specific commands, enter HELP followed by the name of the command for which help is desired.

```
iSQL> HELP;
Use 'help [command]'
Enter 'help index' for a list of command
```

```
iSQL> HELP INDEX;
@          EDIT          QUIT
/          EXIT          ROLLBACK
ALTER      HEADING      SAVE
AUTOCOMMIT H[ISTORY]    SELECT
COMMIT     INSERT       SPOOL
CREATE     LINESIZE      START
DELETE     LOAD         TIMING
DESC       LOBOFFSET    UPDATE
```

DROP	LOBSIZE	VAR [TABLE]
EXECUTE	MOVE	TERM
EXPLAINPLAN	PAGESIZE	VERTICAL

```
iSQL> HELP EXIT;  
exit;  
or  
quit; - exit iSQL
```

# 2.15 Using National Character Sets

When using NCHAR and NVARCHAR type character constants, if the following environment variables settings are made, there will be no concerns over possible data loss.

- The ALTIBASE\_NLS\_NCHAR\_LITERAL\_REPLACE environment variable must be set to 1.

```
$ export ALTIBASE_NLS_NCHAR_LITERAL_REPLACE=1
```

- In order to use NCHAR type data that are encoded differently from the database character set, enter the character “N” in front of the string.

```
iSQL> CREATE TABLE t1 (c1 NVARCHAR(10));  
Create success.
```

```
iSQL> INSERT INTO t1 VALUES (N'AB 가나 ');  
1 row inserted.
```

```
iSQL> SELECT * FROM t1;  
C1  
-----  
AB 가나  
1 row selected.
```

# Index

! 13  
@ 10, 31  
@@ 10, 32  
/ 13

## A

ALTIBASE\_DATE\_FORMAT 15  
ALTIBASE\_HOME 14  
ALTIBASE\_IPC\_FILEPATH 16  
ALTIBASE\_NLS\_NCHAR\_LITERAL\_REPLACE 15, 60  
ALTIBASE\_NLS\_USE 14  
ALTIBASE\_PORT\_NO 14  
AUTOCOMMIT ON/OFF 30

## C

COLSIZE 11, 39  
comment 13  
Configuring iSQL 17  
CONNECT 23

## D

DESC 9, 28  
DISCONNECT 8, 26  
Display Settings 46  
drop procedure 54

## E

ED 10, 33  
EDIT 33  
Edit query statements 10  
Editing the LOGIN file 17  
END 51  
environment variables 14  
EXEC 13, 48  
EXECUTE 13, 48  
Execution time 11  
EXIT 8  
EXPLAIN PLAN 43

## F

FEEDBACK ON/OFF 37  
FOREIGNKEYS ON/OFF 41  
Formatting SELECT result 36  
function 56  
function creation 56  
function execution 56

## G

glogin.sql 17

## H

HEADING ON/OFF 38  
HELP 13, 58  
HISTORY 13, 58  
host variable 13, 48

## I

iSQL 2  
iSQL Command 8  
iSQL Command Line Options 5  
ISQL\_BUFFER\_SIZE 15  
ISQL\_CONNECTION 15  
ISQL\_EDIT 35  
ISQL\_EDITOR 16

## L

LINESIZE 36  
LOAD 33  
load 10  
LOBOFFSET 11, 37  
LOBSIZE 11, 36  
log in 20  
login.sql 17

## N

national character process 60  
NCHAR 60  
NVARCHAR 60

## P

PAGESIZE 38  
performance view 27  
PLANCOMMIT ON/OFF 30  
PREPARE 13  
PREPARE SQL statement 50  
PRINT 13, 49  
procedure creation 51  
procedure execution 51

## Q

QUERYLOGGING ON/OFF 33  
QUIT 8

## R

re-execution 58

Repeat execution 13

## **S**

SAVE 10, 32  
seq 28  
sequence 28  
Sequence information 9  
set foreignkeys 11  
set heading 11  
set linesize 11  
set pagesize 11  
set term 11  
set timing 11  
Setting Up iSQL 3  
Shell command 13  
shell commands 58  
show all 12  
show foreignkeys 12  
show heading 12  
show linesize 12  
show pagesize 12  
show timing 12  
show user 12  
shutdown 8, 21  
SPOOL 10, 31  
START 10  
startup 8, 21  
Stop 8  
stored procedure 51  
SYSDBA 23

## **T**

tab 9, 27  
Table list 9  
Table structure 9  
TERM ON/OFF 42  
TIMESCALE 40  
TIMING ON/OFF 40  
Transaction mode 9  
transaction mode 30

## **V**

VAR 48  
VARIABLE 48  
VERTICAL ON/OFF 45