Altibase® Tools & Utilities

iSQL User's Manual

Release 7.1 (September 18, 2017)

ALTIBASE

Altibase® Tools & Utilities iSQL User's Manual
Release 7.1

Altibase Corp.
10F, Daerung PostTower II,
306, Digital-ro, Guro-gu, Seoul 08378, Korea
Telephone: +82-2-2082-1000 Fax: 82-2-2082-1099
Homepage: http://www.altibase.com

Contents

Preface

About This Manual

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

Intended Audience

The following Altibase 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 7.1 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

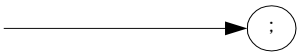
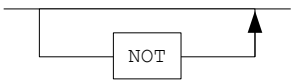
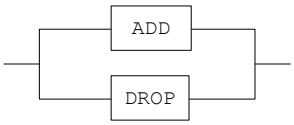
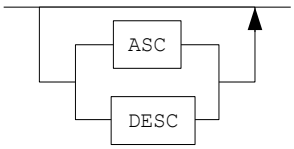
This section describes the conventions used in this manual. Understanding these conventions will make it easier to find information in this manual and other manuals in the series.

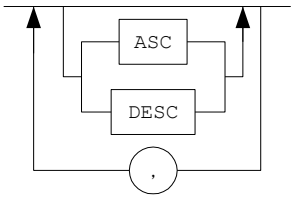
There are two sets of conventions:

- syntax diagrams
- sample code conventions

Syntax Diagrams

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

Elements	Meaning
	Indicates the start of a command. If a syntactic element starts with an arrow, it is not a complete command.
	Indicates that the command continues to the next line. If a syntactic element ends with this symbol, it is not a complete command.
	Indicates that the command continues from the previous line. If a syntactic element starts with this symbol, it is not a complete command.
	Indicates 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
	Indicates an optional element in which multiple elements may be specified. A comma must precede all but the first element.

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

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

On-line Manuals

Online versions of our manuals (PDF or HTML) are available from Altibase's Customer Support site(http://altibase.com/support-center/).

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

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

Thank you. We appreciate your feedback and suggestions.

1 Using iSQL

1.1 iSQL Overview

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

1.1.1 iSQL Main Functionality

Altibase 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 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 Server is running
- port_no
  The port number used when connecting via TCP, IPC or IPCDA
- 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 iSQL Command-Line Options.)

ALTIBASE_HOME environment variable must be set in order to use iSQL. Although it is automatically configured when the server is installed in general, but the user should directly configure since there might be a chance of conflict with the environment variables in the server.

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

```
isql [-H]
     [-S server_name]
     [-PORT port_no]
     [-U user_id] [-P password] [/NOLOG]
     [-SYSDBA]
     [-UNIXDOMAIN-FILEPATH filepath]
     [-IPC-FILEPATH filepath]
     [-SILENT]
     [-F infile_name [param1 [param2]...]] [-O outfile_name] [-NLS_USE nls_name]
     [-NLS_NCHAR_LITERAL_REPLACE 0|1]
     [-prefer_ipv6] [-TIME_ZONE timezone]
     [-ssl_ca CA_file_path | -ssl_capath CA_dir_path]
     [-ssl_cert certificate_file_path]
     [-ssl_key key_file_path]
     [-ssl_verify]
     [-ssl_cipher cipher_list]
```

- -S *server_name*

This option specifies the name (or IP address) of a computer on which Altibase Server is running.

If connection is attempted while the ISQL CONNECTION environment variable is set to IPC or UNIX, and the remote server is specified for this option, iSQL ignores the ISQL CONNECTION specification and connects to the remote server via TCP, and outputs a warning message that the ISQL CONNECTION specification has been ignored. 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 Administrator's Manual*.

- -PORT *port_no*

This option specifies the port number for connecting via TCP, IPC or IPCDA. However, when connecting in a Unix environment via IPC, this option must not be specified. After a warning

message is output, connection to the server is made. To connect via TCP, first set 'ISQL_CONNECTION=TCP' on the client and then enter the PORT_NO.

If the environment variable ISQL_CONNECTION is not set to IPC and the -PORT option is omitted, ALTIBASE_PORT_NO and PORT_NO property is referred in sequence. However, the prompt for port number input is output if all is not specified.

- -U *user_id*

This option specifies a user ID registered in the database.

- -P *password*

Specifies the password corresponding to the user ID.

- /NOLOG

This executes iSQL without connecting to the database.

- -SYSDBA

This allows the SYS user to execute iSQL in SYSDBA mode. If the server has not yet started, iSQL connects as an idle instance and allows the user to start the server.

- -UNIXDOMAIN-FILEPATH *filepath*

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 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 because they have different socket paths. In this case, Unix domain communication can be achived 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.

- -IPCDA-FILEPATH *filepath*

If ALTIBASE_HOME is different from each other when attempting to connect the client and server via IPCDA (ISQL_CONNECTION=IPCDA), the connection cannot be made due to different socket paths Howeve, if ALTIBASE_HOME/trc/cm-ipcda file is used, the Unix domain communication is enabled to bring the information of shared memory. However, this option can be omitted if

IPCDA_FILEPATH of environment variables is specified.

- -F *infile_name* [*param1 [param2]...*]

This command option specifies a script file to be executed immediately after iSQL is launched. Use double quotation marks if the file name contains special characters or spaces.

e.g.)-F \" file name\" e.g.) -F \" file name\"

This command also can specify a paramater value which will be substituted for a substitution variable in the script file. Refer to the 'Passing parameters through START command' for more information regarding the substitution variables.

- -O *outfile_name*

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

Use double quotation marks if the file name contains special characters or spaces.

e.g.)-O \" file name\" e.g.) O \" file name\"

- -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
— MS936
— UTF8
— SHIFTJIS
— MS932

— 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 prefered.
If this option is omitted, isql connects to the IPv4 address by default.
If it fails to connect to the prefered 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.

- -TIME_ZONE timezone

This option sets the time zone of the client. If DB_TZ is specified for this option, the time zone is defaulted to that of the database server. Time zone names like Asia/Seoul, abbreviations such as KST and UTC offset values as +09:00 are valid for specification.
If this option is omitted, the time zone set for the ALTIBASE_TIME_ZONE environment variable is defaulted to the time zone of the client; on omission of the environment variable, the time zone is defaulted to that of the database server.

- -ssl_ca CA_file_path

Specifies the location of the certification authority (CA) certificate in which the public key of the Altibase server to be connected to is incorporated.

- -ssl_capath CA_dir_path

Specifies the directory under which the certification authority (CA) certificate in which the public key of the Altibase server to be connected is incorporated.

- -ssl_cert certificate_file_path

Specifies the location of the client authentication file.

- -ssl_key key_file_path

Specifies the location of the client private key file.

- -ssl_verify

Verifies the certificate the client receives from the server.

- -ssl_cipher cipher_list

Specifies a cipher list for SSL encryption. Please refer to the SSL_CIPHER_LIST property in the *General Reference*.

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 shutdown	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 iSQL Command-Line Options section.
	Prompt	iSQL>	Type a command at the iSQL prompt and press the ENTER key.
	Shutdown	EXIT; QUIT;	Used to shut down iSQL.
Altibase startup and shutdown	Altibase Startup	STARTUP	Use the PRE-PROCESS, PROCESS, CONTROL, META, or SERVICE option to start Altibase up to the corresponding stage.
	Altibase Shutdown	SHUTDOWN	Use one of the NORMAL, IMMEDIATE, or ABORT options to shut down Altibase.
Database connection and disconnection	Access the server as another user	CONNECT [*logon*] [*nls*] [AS sysdba]; where *logon* has the syntax: *user1*[/*pass1*] where *nls* has the syntax: NLS=*character_set*	This command allows access to the database as *user1* with password *pass1* 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 *nls* option specifies the character set. For detailed information on character sets, please refer to the iSQL Command-Line Options: -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 *samp*
	Display sequence Information	SELECT * FROM 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 *seq_name* when EXPLAIN PLAN is ON (or ONLY) and AUTOCOMMIT is OFF. Default: OFF
File management	Saving results to a file	SPOOL file_name ;	Starts writing the results of executed command in iSQL to the file *file_name*.
		SPOOL OFF;	Stops spooling.
	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 *file_name* in the same directory as the calling script.

Category	Type	Commands	Description
	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[IT]	This command edits the most recently executed query.
		ED[IT] *file_name*[.sql]	This command edits existing files or new files.
		2ED[IT] or 2 ED[IT]	This edits the query statements with the number 2 in the history list.
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
		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
		SET FEED[BACK] ON; SET FEED[BACK] OFF; SET FEED[BACK] n;	Determines whether to output the number of rows in a query 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

Category	Type	Commands	Description
		SET HEADING ON; SET HEADING OFF;	Sets whether to output the header of a SELECT result Default: ON
		SET COLSIZE N ;	Sets the number of characters to output when CHAR or VARCHAR type columns are output as a SELECT query result.
		SET NUM[WIDTH] N;	Sets the number of characters to output when data of NUMERIC, DECIMAL, NUMBER, FLOAT type columns are output as a SELECT query result. Default : 11
		CL[EAR] COL[UMNS]	This command releases the column format which has been specified with the COLUMN.
		COL[UMN] [{column | expr} [option]]	This command verifies and configures the display format for a SELECT target column.
		SET NUMF[ORMAT] format;	This command sets the display format of SELECT results of NUMERIC, DECIMAL, NUMBER, and FLOAT type.
	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 CHECK constraint information	SET CHKCONSTRAINTS ON; SET CHKCONSTRAINTS OFF;	Sets whether to output CHECK constraint output including information when displaying the table structure(using DESC). Default: OFF
	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

Category	Type	Commands	Description
	Show/hide partition information	SET PARTITIONS ON; SET PARTITIONS OFF;	Determines whether to include partition information in the output when displaying the table structure (using DESC). Default: OFF
Control output option	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
	Show/hide script commands	SET ECHO ON; SET ECHO OFF;	Sets whether or not to output the commands in the script file that is executed with @, when the script execution results are not output (when TERM option is set to OFF). Default : OFF
	Replace Substitution Variable	SET DEFINE ON; SET DEFINE OFF;	This command specifies whether or not to replace substitution variables with parameter values inserted by a user when executing a script file containing substitution variables. Default : OFF
	Display contents before/after replacing substitution variable	SET VERIFY ON; SET VERIFY OFF;	This command specifies whether or not to display SQL statements before and after the substitution variables are replaced with the parameter values when executing a script file containing substitution variables. Default: ON
	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.

Category	Type	Commands	Description
		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 wil 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 ECHO	Shows the current ECHO 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 CHKCONSTRAINTS	Shows whether the current CHECK constraint is set or not.
		SHOW FOREIGNKEYS	Shows the current foreign key display setting.
		SHOW PARTITIONS	Shows whether the current partition display is set or not.
		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.
Variable and Prepared SQL statements	Variable declaration	VAR *p1* INTEGER;	Declares the variable *p1* as integer type.
		VARIABLE *p2* CHAR(*10*);	Declares the variable *p2* as CHAR type.

Category	Type	Commands	Description
	Assign values to variables	EXECUTE :p1 := 100 ;	Assigns the value *100* to variable *p1*.
		EXEC :p2 := 'abc' ;	Assigns the text *'abc'* to variable *p2*.
	Variable display	PRINT VAR[IABLE] ;	Shows the currently declared variables.
		PRINT *p1*;	Shows the type and value of variable *p1*.
	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.
	Command prompt change	SQLP[ROMPT] {text}	This configures the iSQL command prompt.
	Comment	/* comment */ -- comment	Indicate a multiple-line comment and a single-line comment, respectively.
	Help	HELP; HELP INDEX; HELP EXIT;	This 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

ALTIBASE_HOME is the environment variable which must be configured in order to used iSQL.

Although it is automatically configured when the server is installed in general, but the user should directly configure since there might be a chance of conflict with the environment variables in the server.

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_SSL_PORT_NO

The port number of the server iSQL is to connect to on SSL/TLS.

The -PORT option, environment variables, ALTIBASE_SSL_PORT_NO, the properties in the altibase.properties file take priority in this order as the port number in SSL. On omission, the command prompt asks the user to enter the port number.

1.5.4 ALTIBASE_NLS_USE

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

- US7ASCII
- KO16KSC5601
- MS949
- BIG5
- GB231280
- MS936

- UTF8
- SHIFTJIS
- MS932
- 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.

1.5.5 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.*

1.5.6 ISQL_CONNECTION

When Altibase is operated with a client-server arrangement, the user can select the client-server protocol that is suitable for the operating environment by setting environment variables. Altibase supports the TCP/IP, IPC, IPCDA, and Unix domain SSL socket protocols. The default protocol for communication with Altibase servers is TCP/IP.

Note that when using the IPC or IPCDA protocol the value of Altibase properties related to the IPC channel (IPC_CHANNEL_COUNT or IPCDA_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
```

Note: *If the value set for the ISQL_CONNECTION environment variable is UNIX or IPC, and the remote server is specified for the -s option, a warning message that the setting for ISQL_CONNECTION has been ignored is output and iSQL connects to the remote server using TCP.*

1.5.7 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

1.5.8 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'

1.5.9 ISQL_EDITOR

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

Ex)

CSH: setenv ISQL_EDITOR /usr/bin/ed
SH: ISQL_EDITOR=/usr/bin/ed; export ISQL_EDITOR

1.5.10 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 since they have different Unix domain socket paths. In this case, in order to be able to connect via IPC, it is 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.

1.5.11 IPCDA_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 IPCDA since they have different Unix domain socket paths. In this case, if IPCDA_FILEPATH environment variables or –IPCDA -FILEPATH is specified as a file of $ALTIBASE_HOME/trc/cm-ipcda in the server connection via IPCDA is possible because the server and client can use the identical socket file.

1.5.12 ALTIBASE_TIME_ZONE

This environment variable sets the time zone of the client. If DB_TZ is specified for this option, the time zone is defaulted to that of the database server.

This environment variable can be set with time zone names like Asia/Seoul, abbreviations such as KST and UTC offset values as +09:00 are valid for specification.

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 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.
 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.
```

1.6.4 Notes

For security reasons, the CONNECT command which inputs both the user name and password cannot be used with the LOGIN file. If the CONNECT command is included in the LOGIN file, the following warning message is output and the command is not executed.

WARNING: CONNECT command in glogin.sql file ignored.

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.

-SYSDAB option should be used in order for the SYS user to use iSQL as an administrator. The SYSDBA option can be also used for remote access. Use double quotation marks if the user ID contains special characters or spaces.

```
$ isql -U \"user name\"
```

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 SQL Reference*. For detailed information on errors that may arise during iSQL execution, please refer to the *Altibase 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

iSQL can be used to start up and shut down Altibase.

2.2.1 Starting Up Altibase

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

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

The following is an example of the use of iSQL to start up Altibase. For more information on starting up Altibase, please refer to the *Altibase 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 : ..... [SUC-
CESS]
   [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

Use the SHUTDOWN command to shut down a running Altibase server.

The following is an example of the use of iSQL to shut down Altibase. For more information on shutting down Altibase, please refer to the *Altibase 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 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 *longon* 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.

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) :
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
---------------------------------------------
ALTITBL                                    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
                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.
```

2.3.1.4 Note

Double quotation marks should be used if the name contains special characters or spaces.

```
iSQL> CONNECT "user name";
```

2.3.2 Connecting on SSL

2.3.2.1 Server-Exclusive Mode

When using a private certificate in server-exclusive mode (when the SSL_CLIENT_AUTHENTICATION property is set to 0), the location of the client certificate and private key file need not be specified, as the server does not authenticate the client.

Enable the -ssl_verify option and specify the location of the CA certificate file in which the server public key is incorporated, to verify the certificate received from the server.

```
$ export ISQL_CONNECTION=SSL
$ isql -s localhost -u sys -p MANAGER
or
$ isql -s localhost -u sys -p MANAGER -ssl_verify -ssl_ca ~/cert/ca-cert.pem
```

2.3.2.2 Mutual Authentication Mode

When using a private certificate in mutual authentication mode (when the SSL_CLIENT_AUTHENTICATION property is set to 1), the location of the client certificate and private key file need to be specified, as the server performs client authentication.

Enable the -ssl_verify option and specify the location of the CA certificate file in which the server public key is incorporated, to verify the certificate received from the server.

```
$ export ISQL_CONNECTION=SSL
$ isql -s localhost -u sys -p MANAGER \
-ssl_cert ~/cert/client-cert.pem \
-ssl_key ~/cert/client-key.pem
or
$ isql -s localhost -u sys -p MANAGER \
-ssl_verify -ssl_ca ~/cert/ca-cert.pem \
-ssl_cert ~/cert/client-cert.pem \
-ssl_key ~/cert/client-key.pem
```

2.3.3 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.

```
iSQL> 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.
```

2.3.4 Executing iSQL with the NOLOG Option

The /NOLOG option allows the user to execute iSQL without connecting to the database. The server IP address and port number must be specified to use this option.

```
isql -s localhost -port 20300 /NOLOG
```

Once iSQL is running, enter the database user ID and password with the CONNECT command to connect to the database, and then execute a SQL statement.

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:

```
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 and the meanings of the columns, please refer to the *Altibase 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.

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

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
```

Use double quotation marks if the table name contains special characters or spaces.

```
iSQL> DESC "table name";
iSQL> DESC "user name"."table name";
```

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
-------------------------------------------------------------------------------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 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 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.

```
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).*

2.6.2.3 Passing parameters through SART Command

START file_name[.sql] [param1 [param2] ...]

@file_name[.sql] [param1 [param2] ...]

@@file_name[.sql] [param1 [param2] ...]

[param1 [param2] ...] : The value to be transferred as a parameter to the script file.

The substitution variables are used if a user wants to specify every time execution is made and not fixating certain values of a SQL statement within the script file. The values to be replaced the substitution variable can be passed as a parameter if the script file is executed with START, @ or @@ command. The substitution variable within the script file is used with '&' and numbers, and the number signifies the sequence. However, this feature is performed only if the SET DEFINE ON option is specified. Refer to the SET DEFINE(hyperlink) for further information.

For instance, if substitution variables are used in emp.sql file as in the following :

```
SELECT ENO, E_LASTNAME FROM EMPLOYEES
WHERE EMP_JOB = '&1'
AND SALARY > &2;
```

If 'programmer' and '2000' are inserted as parameters when executing the START command, 'programmer' and '2000' are replaced into &1 and &2, respectively. Thus, employees whose job is 'programmer' and salary is '2000' are viewed.

```
iSQL> SET DEFINE ON; -- Substitution values are replaced as parameters if it is set to ON.
iSQL> START emp.sql programmer 2000
old     2: WHERE EMP_JOB = '&1'
new     2: WHERE EMP_JOB = 'programmer'
old     3: AND SALARY > &2;
new     3: AND SALARY > 2000;

ENO               E_LASTNAME
------------------------------------
10                Bae
```

iSQL outputs SQL commands before and after parameter values are replaced for the command-lines containing substitution variables. SQL commands after replacing values are not output if the SET VERIFY OFF option is specified. The substitution variable can be used for multiple times within a single script and it is not necessary to be used in sequence.

The substitution value can also be replaced with parameters in the following manner.

```
START emp.sql
...
Enter value for 1: programmer
old     2: WHERE EMP_JOB = '&1'
new     2: WHERE EMP_JOB = 'programmer'
Enter value for 2: 2000
old     3: AND SALARY > &2;
new     3: AND SALARY > 2000;
```

In addition, in order to use specific characters by connecting immediately after the substitution value, a period(.) should be used for distinguishing the substitution value and characters.

```
SELECT E_LASTNAME FROM EMPLOYEES WHERE ENO='&1.0';
Enter value for 1: 2
old     1: SELECT E_LASTNAME FROM EMPLOYEES WHERE ENO='&1.0';
new     1: SELECT E_LASTNAME FROM EMPLOYEES WHERE ENO='20';
```

2.6.2.4 SET DEFINE {ON|OFF}

This specifies whether or not to replace substitution variables with the parameter values inserted by a user when executing a script file containing the substitution variable through the START, @ or @@ command.

The default value is set to OFF, and substitution variables are not replaced with parameter values. That is, this option should be set to ON when executing a script file containing substitution variables.

2.6.2.5 SET VERIFY {ON|OFF}

This specifies whether or not to display SQL statements before and after replacing with the parameter value when executing a script file containing the substitution variable through the START, @ or @@ command.

The default value is set to ON and before and after SQL statements are output.

```
$cat Param1.sql
SELECT * FROM T1 WHERE I1 = &1;


iSQL> SET DEFINE ON;
```

```
iSQL> SHOW VERIFY;
Verify : On
iSQL> START Param1.sql 5;
iSQL> SELECT * FROM T1
WHERE I1 = &1;
old     2: WHERE I1 = &1;
new     2: WHERE I1 = 5;
T1.I1           T1.I2
----------------------------
5                Hyacinth
1 row selected.

iSQL> SET VERIFY OFF;
iSQL> SHOW VERIFY;
Verify : Off
iSQL> START Param1.sql 5;
iSQL> SELECT * FROM T1
WHERE I1 = &1;
T1.I1           T1.I2
----------------------------
5                Hyacinth
1 row selected.
```

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 edit 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 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, GENDER) VALUES(21, 'Shiloh',
'Reynolds', 'F');
INSERT INTO employees(ENO, E_FIRSTNAME, E_LASTNAME, GENDER, JOIN_DATE) VALUES(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.
```

2.6.7 Note

Use double quotation marks if the file name contains special characters or spaces.

```
iSQL> SPOOL "file name.txt";
iSQL> START "file name.sql";
iSQL> EDIT "file name.sql";
```

2.7 Formatting SELECT Query Results

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

2.7.1 CL[EAR] COL[UMNMS]

This command releases the display format of all of the columns which have been specified by COLUMN commands.

2.7.1.1 Syntax

CL[EAR] COL[UMNS]

2.7.2 COLUMN

This command verifies or sets the display format for a target column of SELECT. The setting is applied to the following cases only.

- The length of character data type.
- The display format of numeric data type.

2.7.2.1 Syntax

COL[UMN] [{*column* | *expr*} [*option*]]

column or expr should be indicating a target column or an expression, and it should be identical with the one used in the SELECT statement.

Every specified column or the specified format can be confirmed by the COL[UMN] [{*column* | *expr*}] command.

The followings can be used in *option.*.

Option	Description
CLE[AR]	This option releases a specified column.
FOR[MAT] *format*	This option sets the display format for the specified column. - The character data type column: This type can set the display length of the CHAR and VARCHAR type. It will take precedence over SET COLSIZE settings. - The numeric data type column: The display format of the NUMBER, DECIMAL, FLOAT, and NUMERIC type can be specified by this option.

Option	Description
	Refer to "General Reference > Data Types > Numeric Data Types > Numeric " for the available format which can be applied into. It will take precedence over SET NUMFORMAT settings.
ON | OFF	This option confirms whether or not to apply the specified display. - OFF: OFF leaves the column setting as it is, however; it is not applied to output - ON: The specified setting is applied.

2.7.2.2 Description

The display format of a target column in the SELECT statement can be specified. If multiple display formats are selected, the last format will be applied.

In order to release the display format, the user can use the CLEAR or OFF option. The differences between the CLEAR and OFF option is that the CLEAR option can completely remove the specified display setting whereas the OFF option executes the same except it is not applied to output.

2.7.2.3 Example

The following example demonstrates displaying the length of an address column in VARCHAR(60) with 20.

```
iSQL> @schema.sql
iSQL> COLUMN address FORMAT A20
iSQL> select cno, address from customers;
CNO                  ADDRESS
--------------------------------------------
1                    2100 Exposition Boul
                     evard Los Angeles US
                     A
...
```

The following commands should be taken in order to delete the given setting.

```
iSQL> COLUMN address CLE
```

2.7.3 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        GENDER
--------------------------------------------------------------------------
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.
```

2.7.4 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.
```

2.7.5 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.
```

2.7.6 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        GENDER
-------------------------------------------------------------------------
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
2 rows selected.
```

2.7.7 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.
```

2.7.8 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.

2.7.9 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.

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

```
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.

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.
```

2.7.10 SET NUM[WIDTH]

This command sets the number of characters to display for data of NUMERIC, DECIMAL, NUMBER and FLOAT columns in SELECT result sets. Data with many significant digits can be made more legible by setting this value high.

The following example sets NUMWIDTH to 30, and then queries NUMERIC, DECIMAL, NUMBER and FLOAT columns.

```
iSQL> CREATE TABLE t1
(
c_numeric NUMERIC(38, 0),
c_decimal DECIMAL(38, 0),
c_number NUMBER(38, 0),
c_float FLOAT(38)
);
Create success.
iSQL> INSERT INTO t1 VALUES(12345678901234567890, 12345678901234567890,
12345678901234567890, 12345678901234567890);
1 row inserted.

iSQL> SET NUMWIDTH 30
iSQL> SELECT c_numeric, c_decimal, c_number, c_float FROM t1;
C_NUMERIC C_DECIMAL
------------------------------------------------------------------
C_NUMBER C_FLOAT
------------------------------------------------------------------
12345678901234567890 12345678901234567890
12345678901234567890 12345678901234567890
1 row selected.
```

2.7.11 SET NUMF[ORMAT]

2.7.11.1 Syntax

SET NUMF[ORMAT] *format*;

2.7.11.2 Description

This command sets a format of NUMERIC, DECIMAL, NUMBER, and FLOAT type to display their SELECT results. It will take precedence over SET NUMWIDTH settings.

Refer to the "General Reference> Data Types > Numeric Data Types > Numeric" in order to grasp on the formatting on format.

The following is an example of viewing through an exponential form.

```
iSQL> create table t1(i1 float(30));
iSQL> insert into t1 values (123456789012);

iSQL> SET NUMFORMAT 9.99EEEE
```

```
iSQL> select * from t1;
T1.I1
--------------
   1.23E+11
1 rows 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
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 Describing CHECK constraints Information

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

```
iSQL> SET CHKCONSTRAINTS ON;            -> Check Constraint information is output.
iSQL> DESC employees;
[ TABLESPACE : SYS_TBS_MEM_DATA ]
[ ATTRIBUTE ]
--------------------------------------------------------------------------------
NAME                                        TYPE                          IS NULL
--------------------------------------------------------------------------------
```

```
ENO                                     INTEGER          FIXED      NOT NULL
E_LASTNAME                               CHAR(20)        FIXED       NOT NULL
E_FIRSTNAME                              CHAR(20)        FIXED       NOT NULL
EMP_JOB                                  VARCHAR(15)     FIXED
EMP_TEL                                  CHAR(15)        FIXED
DNO                                     SMALLINT         FIXED
SALARY                                   NUMERIC(10, 2)  FIXED
GENDER                                   CHAR(1)         FIXED
BIRTH                                   CHAR(6)          FIXED
JOIN_DATE                                DATE            FIXED
STATUS                                   CHAR(1)         FIXED
[ INDEX ]
-------------------------------------------------------------------------------
NAME                                     TYPE      IS UNIQUE      COLUMN
-------------------------------------------------------------------------------
__SYS_IDX_ID_238                         BTREE     UNIQUE         ENO ASC
EMP_IDX1                                 BTREE                    DNO ASC
[ PRIMARY KEY ]
-------------------------------------------------------------------------------
ENO

[ CHECK CONSTRAINTS ]
-------------------------------------------------------------------------------
NAME         : EMP_CHECK_SEX1
CONDITION : GENDER in ('M', 'F')

iSQL> SET CHKCONSTRAINTS OFF;          -> Check Constraint information is not output.
iSQL> DESC employees;
[ TABLESPACE : SYS_TBS_MEM_DATA ]
[ ATTRIBUTE ]
-------------------------------------------------------------------------------
NAME                                     TYPE                        IS NULL
-------------------------------------------------------------------------------
ENO                                     INTEGER          FIXED      NOT NULL
E_LASTNAME                               CHAR(20)        FIXED       NOT NULL
E_FIRSTNAME                              CHAR(20)        FIXED       NOT NULL
EMP_JOB                                  VARCHAR(15)     FIXED
EMP_TEL                                  CHAR(15)        FIXED
DNO                                     SMALLINT         FIXED
SALARY                                   NUMERIC(10, 2)  FIXED
GENDER                                   CHAR(1)         FIXED
BIRTH                                   CHAR(6)          FIXED
JOIN_DATE                                DATE            FIXED
STATUS                                   CHAR(1)         FIXED
```

```
[ INDEX ]
--------------------------------------------------------------------------------
NAME                                     TYPE       IS UNIQUE       COLUMN
--------------------------------------------------------------------------------
__SYS_IDX_ID_238                         BTREE      UNIQUE           ENO ASC
EMP_IDX1                                 BTREE                       DNO ASC
[ PRIMARY KEY ]
--------------------------------------------------------------------------------
ENO
```

2.8.5 Outputting the Execution Results and Commands of Script Files

The SET TERM and SET ECHO commands determine whether or not to output the execution results and commands of script files to the screen.

Script execution results are output(TERM ON) by default. If the TERM option is set to OFF, the commands which are executed and the results that are generated when the script file is executed in iSQL are not output to the screen. Even if the TERM option is set to OFF, however, query results are output to the screen if queries are manually input(e.g., iSQL> select * from t1;). Only when script commands are used(e.g., iSQL> @t.sql), are the results not output to the screen.

Even if the TERM option is set to OFF, the commands executed in the script can be output by setting the ECHO command to ON.

The following example outputs the execution results of a script file.

```
iSQL> SET TERM ON;                -> Outputs the script execution result.
iSQL> @schema.sql
iSQL> ALTER SESSION SET AUTOCOMMIT = TRUE;              ->Beginning of the result.
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 the result.
```

The following example demonstrates how the commands in the script that is executed with @ can be output, although the TERM option is set to OFF, by setting the ECHO option to ON.

```
iSQL> SET TERM OFF; -> The script execution results are not output.
iSQL> @schema.sql
iSQL> SELECT eno, e_firstname, e_lastname FROM employees;
          -> The result is output when the query is manually input.
ENO             E_FIRSTNAME                E_LASTNAME
-------------------------------------------------------------
1               Chan-seung                 Moon
2               Susan                      Davenport
3               Ken                        Kobain
4               Aaron                      Foster
5               Farhad                     Ghorbani
.
.
.
iSQL> SET ECHO ON;    -> Only the commands in the script that is executed with @
are output.
iSQL> @schema.sql
ALTER SESSION SET AUTOCOMMIT = TRUE;
DROP TABLE ORDERS;
DROP TABLE EMPLOYEES;
.
.
.
CREATE INDEX ODR_IDX3 ON ORDERS (GNO ASC);
```

2.8.6 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 execute:

```
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, VARIBALE KEY RANGE and FILTER, are classified and displayed in detail. Therefore, if the WHERE clause is complicated, you can check which predicates 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.

2.8.7 Outputting the partition information

This function allows to view partition information when viewing the table structure with the DESC command.

```
iSQL> create table t1_range(
c1 integer,
c2 integer,
c3 varchar(4))
PARTITION BY RANGE(c3)
(
PARTITION P_2000 VALUES LESS THAN ('2001') TABLESPACE sys_tbs_disk_data,
```

```
PARTITION P_2001 VALUES LESS THAN ('2002') TABLESPACE sys_tbs_mem_data,
PARTITION P_DEFAULT VALUES DEFAULT
) tablespace SYS_TBS_DISK_DATA;

iSQL> SET PARTITIONS ON; -> This command outputs the partition information.
iSQL> DESC t1_range
[ TABLESPACE : SYS_TBS_DISK_DATA ]
[ ATTRIBUTE ]
-------------------------------------------------------
NAME                    TYPE                  IS NULL
-------------------------------------------------------
C1                      INTEGER
C2                      INTEGER
C3                      VARCHAR(4)
T1_RANGE has no index
T1_RANGE has no primary key
[ PARTITIONS ]
-------------------------------------------------------
Method: Range
Key column(s)
-----------------------------------------
NAME
-----------------------------------------
C3
Values
-------------------------------------------------------
PARTITION NAME          MIN VALUE             MAX VALUE
-------------------------------------------------------
P_2000                                         '2001'
P_2001                  '2001'                 '2002'
P_DEFAULT                '2002'
Tablespace
-------------------------------------------------
PARTITION NAME                  TABLESPACE NAME
-------------------------------------------------
P_2000                          SYS_TBS_DISK_DATA
P_2001                          SYS_TBS_MEM_DATA
P_DEFAULT                        SYS_TBS_DISK_DATA
iSQL> SET PARTITIONS OFF;       -> This command does not output the partition information.
iSQL> DESC t1_range
[ TABLESPACE : SYS_TBS_DISK_DATA ]
[ ATTRIBUTE ]
-----------------------------------------------------------------------------------------
NAME                            TYPE                  IS NULL
```

```
--------------------------------------------------------------------------------------------
C1                              INTEGER
C2                              INTEGER
C3                              VARCHAR(4)
T1_RANGE has no index
T1_RANGE has no primary key
```

2.8.8 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
GENDER          : 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 : 100
iSQL> SHOW LOBSIZE
LobSize    : 80
iSQL> SHOW NUMWIDTH
NumWidth : 11
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 CHKCONSTRAINTS
ChkConstraints : Off
iSQL> SHOW FOREIGNKEYS
ForeignKeys : Off
iSQL> SHOW PLANCOMMIT
PlanCommit : Off
iSQL> SHOW QUERYLOGGING
QueryLogging : Off
iSQL> SHOW TERM
Term : On
iSQL> SHOW ECHO
Echo : OFF
iSQL> SHOW FEEDBACK
Feedback : 1
iSQL> SHOW ALL
```

```
User          : SYS
ColSize     : 0
LobOffset : 0
LineSize    : 80
LobSize     : 80
NumWidth : 11
PageSize    : 0
TimeScale : Second
Heading     : On
Timing       : Off
Vertical    : Off
ChkConstraints : Off
ForeignKeys : Off
Partitions : Off
PlanCommit : Off
QueryLogging : Off
Term : On
Echo : Off
Feedback : 1
Fullname : Off
Sqlprompt : "iSQL> "
Define : Off
```

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[INPUT|OUTPUT|INOUTPUT] var_type

The default value is automatically given unless INPUT, OUTPUT or INOUTPUT is specified.

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
```

2.10.1.4 Assigning a Value to a Host Variable

2.10.1.5 Syntax

EXEC[UTE] :var_name := value;

2.10.1.6 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.
```

2.10.1.7 Viewing Host Variables

2.10.1.8 Syntax

PRINT VAR[IABLE]

-> Shows all declared variables.

PRINT var_name

-> Shows the type and value of the variable *var_name*.

2.10.1.9 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 a 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 a SQL statement:

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

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

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 Command Prompt

The prompt can be modified by configuting other values instead of the fundamental command prompt 'iSQL>'. The SET SQLPROMPT dynamically replaces variables when including runtime

variables, such as current accessed user, and current time.

SET SQLP[ROMPT] {text}

The followings are the substitution variables available for use.

Variable	Description
_CONNECT_IDENTIFIER	The connected server. It is expressed with " host:port_no ".
_DATE	The current time. It is expressed through a specified format in the DATE_FORMAT.
_PRIVILEGE	This variable displays the iSQL access privilege. If it is connected with sysdba, '(sysdba)' is replaced.
_USER	The user name currently being connected.

2.14.3.1 Example

```
iSQL>SET SQLPROMPT "_CONNECT_IDENTIFIER> "

iSQL>SET SQLP "_USER> "

iSQL>SET SQLPROMPT "_USER'@'_CONNECT_IDENTIFIER > "

iSQL>SET SQLPROMPT "_USER on _DATE from _CONNECT_IDENTIFIER> "
```

2.14.4 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;
/                   EXIT                PARTITIONS
@                   EXPLAINPLAN         QUERYLOGGING
ALTER               FEEDBACK            QUIT
AUTOCOMMIT          FOREIGNKEYS         ROLLBACK
CHKCONSTRAINTS      FULLNAME            SAVE
CL[EAR]             H[ISTORY]           SELECT
COL[UMN]            HEADING             SPOOL
```

```
COLSIZE          INSERT           SQLP[ROMPT]
COMMIT           LINESIZE         START
CREATE           LOAD             TERM
DEFINE           LOBOFFSET        TIMESCALE
DELETE           LOBSIZE          TIMING
DESC             MERGE            UPDATE
DROP             MOVE             USER
ECHO             NUM[WIDTH]       VAR[IABLE]
EDIT             NUMF[ORMAT]      VERTICAL
EXECUTE          PAGESIZE
```

```
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 to1.

```
$ 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

L

N

P

Q

R

S

T

V