HP 9000 Computer Systems

ALLBASE/ISQL
Reference Manual

HEWLETT PACKARD

Printing History

The following table lists the printings of this document, together with the respective release dates for each edition. The software version indicates the version of the software product at the time this document was issued. Many product releases do not require changes to the document. Therefore, do not expect a one-to-one correspondence between product releases and document editions.

Edition	Date	Product Number	Hardware Series	HP-UX Release
First Edition (800)	March 1987	36217-02A.01.00	800	1.0
First Edition (300)	March 1987	79725 1.0	300	1.0
Update 1 (800)	June 1987	36217-02A.01.03	800	1.3
Update 1 (300)	June 1987	79725 1.0	300	1.0
Update Incorporated (800)	August 1987	36217-02A.01.03	800	1.3
Update Incorporated (300)	August 1987	79725 1.0	300	1.0
Second Edition (800)	February 1988	36217-02A.B2.00.03	800	2.0
Second Edition (300)	February 1988	79725 2.0	300	6.5
Third Edition (800)	November 1988	36217-02A.03.00	800	3.0
Third Edition (300)	November 1988	HP79725 Rev. 6.5	300	6.5
Fourth Edition	September 1989	36217-02A.07.00	800	7.0
		HP79725 Rev. 7.0	300	7.0
Update 1	October 1989	36217-02A.07.00	800	7.0
		HP79725 Rev. 7.0	300	7.0
Fifth Edition	January 1991	36217-02A.E1.00	800	8.0
		36217-02A.E1.00	700	8.0
		HP79725A.E1.00	400	8.0
		HP79725A.E1.00	300	8.0
Sixth Edition	September 1992	36217-02A.F0.00	800	8.0 and 9.0
		36217-02A.F0.00	700	8.0 and 9.0
		HP79725A.F0.00	400	8.0 and 9.0
		HP79725A.F0.00	300	8.0 and 9.0
Seventh Edition	April 1994	36217-02A.G0.00	700/800	9.0

ALLBASE/SQL Manuals

Title	Part Number
ALLBASE/ISQL Reference Manual	36217-90188
ALLBASE/NET User's Guide	36217-90093
ALLBASE/SQL Advanced Application Programming Guide	36217-90186
ALLBASE/SQL C Application Programming Guide	36217-90014
ALLBASE/SQL COBOL Application Programming Guide	36217-90058
ALLBASE/SQL Database Administration Guide	36217-90005
ALLBASE/SQL FORTRAN Application Programming Guide	36217-90013
ALLBASE/SQL Message Manual	36217-90009
ALLBASE/SQL Pascal Application Programming Guide	36217-90007
ALLBASE/SQL Performance and Monitoring Guidelines	36217-90185
ALLBASE/SQL Reference Manual	36217-90001
HP ALLBASE/QUERY User's Guide	92534-64001
HP PC API User's Guide for ALLBASE/SQL	36217-90187
Up and Running with ALLBASE/SQL	36389-90011

Preface

This manual describes ISQL (Interactive Structured Query Language) and how to use it on HP 9000 computers running under the HP-UX operating system. SQL is the language you use to define and maintain data in an ALLBASE/SQL DBEnvironment. ALLBASE/SQL is Hewlett-Packard's proprietary relational database management system.

Most of the examples in this manual are based on the tables, views, and other objects in the sample database environment, PartsDBE, which accompanies ALLBASE/SQL. Refer to appendix C of the *ALLBASE/SQL Reference Manual* for a full description of the sample DBEnvironment. This appendix also contains information on installing and setting up a copy of PartsDBE for practice use.

This manual is for database administrators, application programmers, and any other users who want to execute SQL commands interactively. It presumes that the reader has a grasp of basic relational database concepts and an introductory understanding of ALLBASE/SQL.

This manual contains the following chapters:

- Chapter 1, "Introduction," presents the components of ISQL.

- Chapter 2, "Getting Started with ISQL," describes how to use ISQL.

- Chapter 3, "Using ISQL for Database Tasks," shows how to perform data manipulation and load/unload operations using ISQL.

- Chapter 4, "ISQL Commands," contains an alphabetical reference of all the ISQL commands.

- Appendix A, "SQL Syntax Summary," contains an alphabetical summary of all the SQL commands and other elements of syntax.

- Appendix B, "ISQL Syntax Summary," contains an alphabetical summary of all the ISQL commands.

What's New in this Release

The following table highlights the new or changed functionality in this release, and shows you where each feature is documented.

New Features in ALLBASE/SQL Release G.0

Feature (Category)	Description	Documented in . . .
Stored procedures (Usability)	Provides additional stored procedure functionality for application programs. Allows declaration of a procedure cursor and fetching of multiple rows within a procedure to applications. New statement: ADVANCE. Changed syntax: CLOSE, CREATE PROCEDURE, DECLARE CURSOR, DESCRIBE, EXECUTE, EXECUTE PROCEDURE, FETCH, OPEN.	*ALLBASE/SQL Reference Manual*, "SQL Statements" and "Using Procedures" in "Constraints, Procedures and Rules;" *ALLBASE/SQL Advanced Application Programming Guide*, "Using Procedures in Application Programs."
Case insensitivity (Usability)	Adds an optional attribute to the character and varchar type column attributes of tables. Allows search and compare of these columns in a case insensitive manner. Four new SQLCore data types are added. Changed syntax: ALTER TABLE, CREATE TABLE.	*ALLBASE/SQL Reference Manual*, "Comparison Predicate" in "Search Conditions," CREATE TABLE in "SQL Statements."
Support for 1023 columns (Usability)	Increases the maximum number of columns per table or view to 1023. Increases maximum sort columns and parameters in a procedure to 1023.	*ALLBASE/SQL Reference Manual*, CREATE TABLE and CREATE VIEW in "SQL Statements;" *ALLBASE/SQL Database Administration Guide*, "ALLBASE/SQL Limits" appendix.
ISQL HELP improvements (Usability)	Gives help for entire command instead of only the verb.	*ALLBASE/ISQL Reference Manual*, HELP in "ISQL Commands."
EXTRACT command (Usability)	Extracts modules from the database and stores them in a module file. Allows for creation of a module file at any time based on the current DBEnvironment without preprocessing. New command: EXTRACT. Changed syntax: INSTALL.	*ALLBASE/ISQL Reference Manual*, "Using Modules" in "Using ISQL for Database Tasks," EXTRACT, INSTALL in "ISQL Commands."

New Features in ALLBASE/SQL Release G.0 (continued)

Feature (Category)	Description	Documented in . . .
New SQLGEN GENERATE parameters (Usability)	Generates SQL statements necessary to recreate modified access plans for module sections. New syntax for GENERATE: DEFAULTSPACE, MODOPTINFO, PARTITION, PROCOPTINFO, SPACEAUTH.	*ALLBASE/SQL Database Administration Guide*, "SQLGEN Commands" appendix.
Row level locking (Usability)	Permits multiple transactions to read and update a table concurrently because locking is done at row level. Since the transaction will obtain more locks, the benefits must be weighed against the costs. (Previously documented in an addendum after F.0 release.)	*ALLBASE/SQL Reference Manual*, "Concurrency Control through Locks and Isolation Levels;" *ALLBASE/SQL Database Administration Guide*, "Effects of Page and Row Level Locking" in "Physical Design."
Increased number of users (Usability)	Removes the limitation of 240 users supported by pseudotables. (Maximum is system session limits: 2000 on HP-UX; 1700 on MPE/iX.)	*ALLBASE/SQL Database Administration Guide*, "ALLBASE/SQL Limits" appendix.
POSIX support (Usability)	Improves application portability across MPE/iX and HP-UX. Enhances the ALLBASE/SQL preprocessors to run under POSIX (Portable Operating System Interface) on MPE/iX.	*ALLBASE/SQL Advanced Application Programming Guide*, "POSIX Preprocessor Invocation" in "Using the Preprocessor."
Application thread support (Performance, Usability)	Provides the use of threads in an application. Allows ALLBASE/SQL to be used in an application threaded environment on MPE/iX. Application threads are light weight processes that share some resources and last for the duration of a transaction. Threaded applications reduce the overhead of context switching and improve the performance of OpenTP applications.	*ALLBASE/SQL Advanced Application Programming Guide*, "Using the Preprocessor."

New Features in ALLBASE/SQL Release G.0 (continued)

Feature (Category)	Description	Documented in . . .
High Availability	Provides a collection of features to keep systems available nonstop including: Partial STORE and RESTORE, Partial rollforward recovery, DBEFiles in different groups (MPE/iX), detaching and attaching database objects, CHECKPOINT host variable, changing log files, console messages logged to a file, generating fewer log records by using TRUNCATE TABLE to delete rows, and new system catalog information. See the following features for new and changed syntax.	*ALLBASE/SQL Reference Manual*, "SQL Statements;" *ALLBASE/SQL Database Administration Guide*, "Maintaining a Nonstop Production System" in "Maintenance" chapter and "SQLUtil" appendix.
Partial rollforward recovery (High Availability)	Supports partial rollforward recovery through PARTIAL option on SETUPRECOVERY. Used to recover specific DBEFiles while allowing access to other DBEFiles.	*ALLBASE/SQL Database Administration Guide*, "Backup and Recovery" chapter and SETUPRECOVERY PARTIAL in "SQLUtil" appendix.
Partial STORE and RESTORE (High Availability)	Gives more flexibility in backup and recovery strategies by allowing partial store and restore of DBEFiles, DBEFileSets or combinations of both. See "New and changed SQLUtil commands for increased availability" later in this table.	*ALLBASE/SQL Database Administration Guide*, "Backup and Recovery" chapter and "SQLUtil" appendix.
DBEFile group change on MPE/iX (High Availability)	Manages DBEFiles so they can be placed in a particular group or on a particular volume (MPE/iX). Use either CREATE DBEFILE or MOVEFILE.	*ALLBASE/SQL Reference Manual*, CREATE DBEFile in "SQL Statements;" *ALLBASE/SQL Database Administration Guide*, "Maintaining a Nonstop Production System" in "Maintenance" chapter and MOVEFILE in "SQLUtil" appendix.
Detaching and attaching database objects (High Availability)	Detaches or attaches a DBEFile or DBEFileSet from the DBEnvironment. This is useful for data that is accessed infrequently such as tables containing historical data only. New SQLUtil commands: DETACHFILE, ATTACHFILE.	*ALLBASE/SQL Database Administration Guide*, "Maintaining a Nonstop Production System" in "Maintenance" chapter and DETACHFILE, ATTACHFILE in "SQLUtil" appendix.

New Features in ALLBASE/SQL Release G.0 (continued)

Feature (Category)	Description	Documented in . . .
New and changed SQLUtil commands for increased availability (High Availability)	Adds support for high availability and System Management Intrinsics. Intended for non-stop, continuously available operations. New SQLUtil commands: ATTACHFILE, CHANGELOG, DETACHFILE, RESTORE PARTIAL, STORE PARTIAL, STOREINFO, STOREONLINE PARTIAL, WRAPDBE. Modified SQLUtil commands: MOVEFILE, RESTORE, RESTORELOG, SHOWDBE, SETUPRECOVERY, STORE, STORELOG, STOREONLINE.	*ALLBASE/SQL Database Administration Guide*, "SQLUtil" appendix.
List files on backup device (High Availability)	Lists physical names of files stored on backup device with new SQLUtil command: STOREINFO.	*ALLBASE/SQL Database Administration Guide*, "Backup and Recovery" chapter and STOREINFO in "SQLUtil" appendix.
Log file improvements (High Availability)	Allows changing log files, switching of console messages to a file, and gives advance warning for log full. Increased maximum size of a single DBE log file to 4 gigabytes. A DBEnvironment can have up to 34 log files configured. Changed syntax: CHECKPOINT. New SQLUtil command: CHANGELOG.	*ALLBASE/SQL Reference Manual*, CHECKPOINT in "SQL Statements;" *ALLBASE/SQL Database Administration Guide*, "Maintaining a Nonstop Production System" in "Maintenance" chapter, CHANGELOG in "SQLUtil" appendix, and "ALLBASE/SQL Limits" appendix.
New SET SESSION and SET TRANSACTION statements (Standards, Performance)	Provides additional flexibility and improved performance. Allows setting and changing transaction and session attributes.	*ALLBASE/SQL Reference Manual*, SET SESSION and SET TRANSACTION in "SQL Statements."
FIPS flagger (Standards)	Meets Federal Information Processing Standard (FIPS) 127.1 flagger support. Flags non-standard statement or extension. Invoked with a flagger option in the preprocessor command line or the SET FLAGGER command in ISQL. Updatability rules are different when flagger is invoked. New syntax: DECLARE CURSOR, WHENEVER. Changes to C and COBOL host variable declaration.	*ALLBASE/SQL Reference Manual*, DECLARE CURSOR in "SQL Commands" and "Standards Flagging Support" appendix; *ALLBASE/SQL Advanced Application Programming Guide*, "Flagging Non-Standard SQL with the FIPS Flagger;" *ALLBASE/ISQL Reference Manual*, SET in "ISQL Commands."

New Features in ALLBASE/SQL Release G.0 (continued)

Feature (Category)	Description	Documented in . . .
Optimizer enhancement (Performance)	Uses a more efficient algorithm that significantly reduces the time to generate the access plan.	*ALLBASE/SQL Performance and Monitoring Guidelines*, "Optimization" in "Basic Concepts in ALLBASE/SQL Performance."
Access plan modification (Performance)	Allows modification of access plans for stored section to optimize performance. View the plan with SYSTEM.SETOPTINFO. New statement: SETOPT.	*ALLBASE/SQL Reference Manual*, SETOPT in "SQL Statements;" *ALLBASE/SQL Database Administration Guide*, SYSTEM.SETOPINFO in "System Catalog."
Syntax added to disable access plan optimization (Performance, Usability)	Specifies that the optimization information in the module file is not to be used. Changed syntax: EXTRACT, INSTALL, VALIDATE.	*ALLBASE/SQL Reference Manual*, VALIDATE in "SQL Statements; *ALLBASE/ISQL Reference Manual*," EXTRACT, INSTALL in "ISQL Commands."
Application Development Concurrency (Performance, Usability)	Provides enhancements to improve preprocessing performance when simultaneously accessed by multiple users. Page or row level locking on any system base table and processing without storing sections. See the related features in this table. New SQL parameter: SET DEFAULT DBEFileSet. SQL changed syntax: ALTER TABLE, GRANT, REVOKE, UPDATE STATISTICS. ISQL changed syntax: INSTALL. Changed SYSTEM and CATALOG view. New STOREDSECT tables. Special owners HPRDBSS and STOREDSECT. Changed syntax for Full Preprocessing Mode.	*ALLBASE/SQL Reference Manual*, "Names" and "SQL Statements;" *ALLBASE/SQL Advanced Application Programming Guide*, "Using the Preprocessor;" *ALLBASE/ISQL Reference Manual*, "ISQL Commands;" *ALLBASE/SQL Database Administration Guide*, "Database Creation and Security" and "System Catalog."
System Catalog tables (Performance)	Provides greater concurrency by allowing users to specify table, page, or row level locking of any system table owned by STOREDSECT through the ALTER TABLE statement.	*ALLBASE/SQL Reference Manual*, "Names;" *ALLBASE/SQL Database Administration Guide*, "System Catalog."
Preprocessors (Performance)	Allows optional specification of a DBEFileSet for storage of sections. Allows preprocessing without storing sections in DBEnvironment.	*ALLBASE/SQL Advanced Application Programming Guide*, "Using the Preprocessor."

New Features in ALLBASE/SQL Release G.0 (continued)

Feature (Category)	Description	Documented in ...
I/O performance improvement (Performance)	Optimizes I/O for initial load, index build, serial scans, internal data restructuring, file activity, pseudo mapped files and temporary files. See the following features for new and changed syntax.	*ALLBASE/SQL Reference Manual*, "SQL Statements."
TRUNCATE TABLE statement (Performance)	Deletes all rows in a specified table leaving its structure intact. Indexes, views, default values, constraints, rules defined on the table, and all authorizations are retained. TRUNCATE TABLE is faster than the DELETE statement and generates fewer logs. New statement: TRUNCATE TABLE.	*ALLBASE/SQL Reference Manual*, TRUNCATE TABLE in "SQL Statements."
New scans (Performance)	Reads tables with a new parallel sequential scan. The tables are partitioned and files are read in a round robin fashion to allow OS prefetch to be more effective. Allows the I/O for a serial scan to spread across multiple disc drives.	*ALLBASE/SQL Performance and Monitoring Guidelines*, "Using Parallel Serial Scans" in "Guidelines on Query Design."
Load performance improvement (Performance)	Improves performance with new SET and SET SESSION attributes, a new binary search algorithm, and deferred allocation of HASH pages. New attributes for SET SESSION statement: FILL, PARALLEL FILL.	*ALLBASE/SQL Reference Manual*, SET SESSION in "SQL Statements."
ISQL enhanced to improve the performance of LOADs (Performance)	Uses new parameters of the ISQL SET command to set load buffer size and message reporting. Improves load performance. Choose a procedure, command file, or new ISQL command to set constraints deferred, lock table exclusively, and set row level DML atomicity. Changed syntax: SET (see the following feature).	*ALLBASE/ISQL Reference Manual*, SET in "ISQL Commands."

New Features in ALLBASE/SQL Release G.0 (continued)

Feature (Category)	Description	Documented in ...
Modified SET options (Performance)	Provides better performance for LOADs and UNLOADs. Specify buffer size, status reporting for LOAD/UNLOAD or exclusive lock for data table. AUTOSAVE row limit increased to 2147483647. New and changed SET options: LOAD_BUFFER, LOAD_ECHO, AUTOLOCK, AUTOSAVE.	*ALLBASE/ISQL Reference Manual*, SET in "ISQL Commands;" *ALLBASE/SQL Performance and Monitoring Guidelines*, "Initial Table Loads" in "Guidelines on Logical and Physical Design."
SQLMON (Tools)	Monitors the activity of ALLBASE/SQL DBEnvironment. Provides information on file capacity, locking, I/O, logging, tables, and indexes. Summarizes activity for entire DBEnvironment or focuses on individual sessions, programs, or database components. Provides read-only information.	*ALLBASE/SQL Performance and Monitoring Guidelines*, chapters 6-9.
Audit (Tools)	Provides a series of features to set up an audit DBEnvironment which generates audit log records that you can analyze with the new SQLAudit utility for security or administration. Includes the ability to set up partitions. See *ALLBASE/SQL Database Administration Guide* for SQLAudit commands. Modified statements: ALTER TABLE, CREATE TABLE, START DBE NEW, START DBE NEWLOG. New statements: CREATE PARTITION, DROP PARTITION, DISABLE AUDIT LOGGING, ENABLE AUDIT LOGGING, LOG COMMENT.	*ALLBASE/SQL Reference Manual*, "SQL Statements;" *ALLBASE/SQL Database Administration Guide*, "DBEnvironment Configuration and Security" chapter and "SQLAudit" appendix.
Wrapper DBEnvironments (Tools)	Creates a DBEnvironment to wrap around the log files orphaned after a hard crash of DBEnvironment. New SQLUtil command: WRAPDBE.	*ALLBASE/SQL Reference Manual*, "Wrapper DBEnvironments" in "Using ALLBASE/SQL;" *ALLBASE/SQL Database Administration Guide*, WRAPDBE in "SQLUtil."
HP PC API is now bundled with ALLBASE/SQL.	PC API is an application programming interface that allows tools written with either the GUPTA or the ODBC interface to access ALLBASE/SQL and IMAGE/SQL from a PC.	*HP PC API User's Guide for ALLBASE/SQL*.

New Features in ALLBASE/SQL Release G.0 (continued)

Feature (Category)	Description	Documented in . . .
Increased memory for MPE/iX (HP-UX shared memory allocation is unchanged) (Performance)	Increases memory up to 50,000 data buffer pages and 2,000 run time control block pages. Increases the limits significantly allowing allocation of enough data buffer pages to keep the entire DBEnvironment in memory if desired for performance.	*ALLBASE/SQL Reference Manual*, STARTDBE, STARTDBE NEW, and START DBE NEWLOG in "SQL Statements;" *ALLBASE/SQL Database Administration Guide*, "ALLBASE/SQL Limits" appendix.
ALLBASE/NET enhancements (Connectivity, Performance)	Improves performance of ALLBASE/NET, allows more client connections on server system, and reduces number of programs on MPE/iX.	*ALLBASE/NET User's Guide*, "Setting up ALLBASE/NET."
ALLBASE/NET commands and options for MPE/iX (Connectivity, Usability)	Adds option ARPA. Adds option NUMSERVERS to check status of listeners and number of network connections. Changed syntax: ANSTART, ANSTAT, ANSTOP. Changed NETUtil commands: ADD ALIAS, CHANGE ALIAS.	*ALLBASE/NET User's Guide*, "Setting up ALLBASE/NET" and "NETUtil Reference."
ALLBASE/NET and NetWare (Connectivity)	ALLBASE/NET listener for NetWare now works with the 3.11 version of Novell's NetWare for UNIX (HP NetWare/iX).	*ALLBASE/NET User's Guide*, "Setting up ALLBASE/NET."
Changed restrictions for executing NETUtil commands for MPE/iX (Connectivity, Usability)	Adds SM or AM (in the specified account) to MANAGER.SYS for adding, changing, or deleting users for MPE/iX.	*ALLBASE/NET User's Guide*, "Setting up ALLBASE/NET."
ARPA is only TCP/IP interface for data communication through ALLBASE/NET beginning with HP-UX 10.0 (Connectivity)	Remote database access applications that specify NS will not work if the client and/or server machine is an HP 9000 Series 700/800 running HP-UX 10.0 or greater. Server Node Name entry must be changed from NS node name to ARPA host name. For the NETUsers file, the "Client Node Name" must be changed from the NS node name to the ARPA host name. New NETUtil commands: MIGRATE USER, MIGRATE ALIAS.	*ALLBASE/NET User's Guide*, "Setting up ALLBASE/NET" and "NETUtil Reference."

Conventions

UPPERCASE   In a syntax statement, commands and keywords are shown in
                uppercase characters. The characters must be entered in the order
                shown; however, you can enter the characters in either uppercase or
                lowercase. For example:

        `COMMAND`

        can be entered as any of the following:

        `command`   `Command`   `COMMAND`

        It cannot, however, be entered as:

        `comm`   `com_mand`   `comamnd`

italics   In a syntax statement or an example, a word in italics represents a
                parameter or argument that you must replace with the actual value.
                In the following example, you must replace *filename* with the name
                of the file:

        `COMMAND` *filename*

punctuation   In a syntax statement, punctuation characters (other than brackets,
                braces, vertical bars, and ellipses) must be entered exactly as shown.
                In the following example, the parentheses and colon must be entered:

        (*filename*):(*filename*)

underlining   Within an example that contains interactive dialog, user input and
                user responses to prompts are indicated by underlining. In the
                following example, yes is the user's response to the prompt:

        `Do you want to continue? >>` `yes`

{   }   In a syntax statement, braces enclose required elements. When
                several elements are stacked within braces, you must select one. In
                the following example, you must select either `ON` or `OFF`:

$$\texttt{COMMAND} \begin{Bmatrix} \texttt{ON} \\ \texttt{OFF} \end{Bmatrix}$$

[   ]   In a syntax statement, brackets enclose optional elements. In the
                following example, `OPTION` can be omitted:

        `COMMAND` *filename* `[OPTION]`

                When several elements are stacked within brackets, you can select
                one or none of the elements. In the following example, you can select
                `OPTION` or *parameter* or neither. The elements cannot be repeated.

$$\texttt{COMMAND}\ \textit{filename}\ \begin{bmatrix} \texttt{OPTION} \\ \textit{parameter} \end{bmatrix}$$

Conventions (continued)

[...]   In a syntax statement, horizontal ellipses enclosed in brackets indicate that you can repeatedly select the element(s) that appear within the immediately preceding pair of brackets or braces. In the example below, you can select *parameter* zero or more times. Each instance of *parameter* must be preceded by a comma:

[,*parameter*][...]

In the example below, you only use the comma as a delimiter if *parameter* is repeated; no comma is used before the first occurrence of *parameter*:

[*parameter*][,...]

| ... | In a syntax statement, horizontal ellipses enclosed in vertical bars indicate that you can select more than one element within the immediately preceding pair of brackets or braces. However, each particular element can only be selected once. In the following example, you must select **A, AB, BA**, or **B**. The elements cannot be repeated.

$$\begin{Bmatrix} A \\ B \end{Bmatrix} | \ \ ... \ |$$

...   In an example, horizontal or vertical ellipses indicate where portions of an example have been omitted.

Δ   In a syntax statement, the space symbol Δ shows a required blank. In the following example, *parameter* and *parameter* must be separated with a blank:

(*parameter*)Δ(*parameter*)

⬭   The symbol ⬭ indicates a key on the keyboard. For example, (RETURN) represents the carriage return key or (Shift) represents the shift key.

(CTRL)*character*   (CTRL)*character* indicates a control character. For example, (CTRL)Y means that you press the control key and the Y key simultaneously.

Contents

Figures

Tables

1

Introduction

ISQL stands for **Interactive Structured Query Language**. With ISQL, you can *interactively* define and access data in an ALLBASE/SQL relational database environment. This manual describes the ISQL commands and how to use ISQL to execute SQL and ISQL commands. The following topics are discussed in this chapter:

- Purpose of ISQL.
- ISQL Users.
- Authorization.
- Native Language Support.

Purpose of ISQL

You use ISQL to enter SQL commands at a terminal. **SQL** is the language used to define and access an ALLBASE/SQL database. Most SQL statements can be submitted through ISQL, although a few SQL statements specifically support programmatic database access and can only be submitted in application programs. As Figure 1-1 illustrates, ISQL interacts with **SQLCore**, one of the components of ALLBASE/SQL. SQLCore checks the syntax of statements and prepares them for processing.
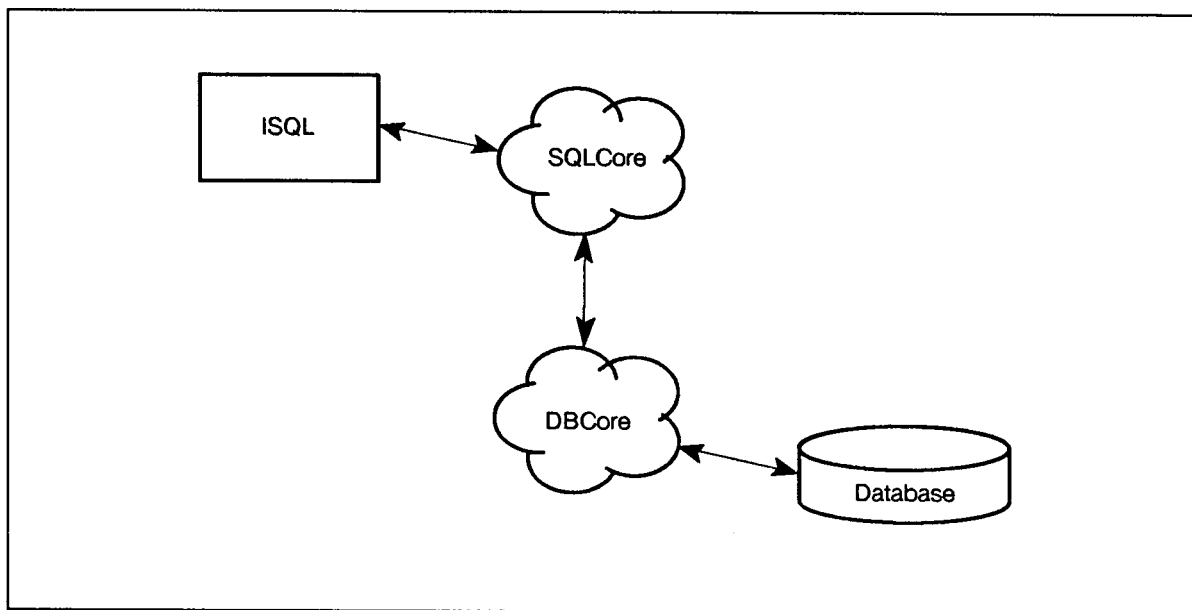
LG200142_011

Figure 1-1. ISQL and SQLCore Interaction

ISQL also has its own set of commands which facilitate the interactive use of SQL statements. For example, you can use ISQL commands to quickly change and resubmit previously executed SQL statements. You can also group SQL statements together in a file and execute the file through ISQL.

From ISQL, you can use ISQL and SQL commands to do the following:

- Create and maintain DBEnvironments.
- Manipulate data.
- Load and unload data.
- Install modules.

Note   ALLBASE/SQL users on diskless machines cannot create a DBEnvironment. The DBEnvironment must be created by a user on the DUX cluster server. The DBECreator can then give the diskless user access to the DBEnvironment by setting up ALLBASE/NET and granting the appropriate authorities. See the *ALLBASE/NET User's Guide* for additional information about DUX cluster operation.

All SQL statement operate on a DBEnvironment; some of the ISQL commands also result in DBEnvironment access. ISQL uses SQLCore to carry out all DBEnvironment access operations.

The *ALLBASE/SQL Reference Manual* describes the SQL statement. A section entitled "Scope" is included for each statement to indicate whether a statement can be used in ISQL or an application program. For the use of SQL statements in application programs, refer to the ALLBASE/SQL application programming guide for the language of your choice.

SQLUtil and SQLGEN can be invoked either directly or from ISQL. Refer to the *ALLBASE/SQL Database Administration Guide* for complete information on these utilities.

ISQL Users

ISQL is a useful facility for many types of ALLBASE/SQL users:

- Application programmers who use ISQL as a program development tool:

  - To obtain information about table and view structures.
  - To try out queries to be embedded in applications.
  - To build test databases for debugging data manipulation operations and for testing programs.

- Database administrators who use ISQL as a DBA tool:

  - To create, build, and maintain databases.
  - To define data and conduct authorization operations as required.
  - To query system catalog views to monitor DBEnvironment usage.
  - To load or unload tables, and install or drop preprocessed modules.
  - To define and maintain DBEFiles and DBEFileSets to manage space.

  These users invoke SQLUtil from ISQL for DBEnvironment maintenance activities.

- End users who use ISQL to retrieve or modify data.

Authorization

Special HP-UX capabilities are not required to invoke ISQL. To execute database access statements, however, you must have the proper ALLBASE/SQL authorization. The authorization required to execute SQL statements is defined in the *ALLBASE/SQL Reference Manual*. The authorization required to execute ISQL commands that access a database is defined in this manual. You can access files in directories other than your home directory if their HP-UX file mode permits.

Native Language Support

ALLBASE/SQL lets you manipulate databases in a wide variety of native languages in addition to the default language, known as **n-computer**. You can use either 8-bit or 16-bit character data, as appropriate for the language you select. In addition, you can always include ASCII data in any database, because ASCII is a subset of each supported character set. The collating sequence for sorting and comparisons is that of the native language selected. A list of supported languages is in /usr/lib/nls/config.

You can use native language characters in the various ways listed here:

■ As character literals.
■ As host variables for CHAR or VARCHAR data (but not variable names).
■ In ALLBASE/SQL object names.
■ In WHERE and VALUES clauses.
■ In file names.

If your system has the proper message files installed, ALLBASE/SQL displays prompts, messages, and banners in the language you select. It also displays dates and time according to local customs. In addition, you can respond to ISQL prompts in the native language selected. However, regardless of the native language used, the syntax of ISQL and SQL commands— including punctuation—remains in ASCII.

In order to use a native language other than the default, you must do the following:

1. Make sure your I/O devices support the character set you wish to use.

2. Set the HP-UX environment variable LANG to the native language (*LanguageName*) you wish to use. For the C shell, use the following command:

        setenv LANG   *LanguageName*

   For the K shell, use the following command (no spaces before or after the equal sign):

        typeset -x LANG=*LanguageName*

   For the Bourne shell, use the following commands:

        LANG = *LanguageName*
        export LANG

   This language then becomes the **current language**. which determines those message and welcome files used when you invoke ISQL. If LANG is not set, the current language is n-computer.

3. Use the LANG=*LanguageName* option of the START DBE NEW command to specify the language when you create a DBEnvironment.

You can use native language characters in the DBEnvironment name. If you do so, you must set the LANG environment variable to the same language before you can connect to the DBEnvironment. To avoid confusion, you are advised to always use the same language for the DBEnvironment that you use in the LANG variable.

Resetting the LANG variable while you are connected to a DBEnvironment has no effect on the current DBE session.

ISQL interacts with two files that contain messages which may vary according to the language selected:

- **hpsqlcat** is the ALLBASE/SQL message catalog. This file contains ISQL and other ALLBASE/SQL messages. The full pathname for this file is

      /usr/lib/nls/n-computer/hpsqlcat

  If you are using native language data, the pathname is /usr/lib/nls/$LANG/hpsqlcat, where $LANG is the user's current language specified in the setenv command. If this file is not available, then the default file, /usr/lib/nls/n-computer/hpsqlcat, is used.

- **isqlwel** is a file containing the ISQL prompt, sign-on banner, and messages to be displayed when you invoke ISQL. The full pathname for this file is:

      /usr/lib/nls/n-computer/isqlwel

  If you are using native language data, the path name for this file is /usr/lib/nls/$LANG/isqlwel, where $LANG is the user's current language specified in the setenv command. If this file is not available, then the default file /usr/lib/nls/n-computer/isqlwel is used.

Reading More about ISQL

- Chapter 2 examines how and when the ISQL commands are used.

- Chapter 3 explains how to use ISQL to change and display data in the DBEnvironment.

- Chapter 4 describes all the ISQL commands. It defines their syntax and semantics and provides examples.

- Appendix A contains a summary of SQL command syntax.

- Appendix B contains a summary of ISQL command syntax.

The *ALLBASE/SQL Message Manual* describes how to interpret and handle ALLBASE/SQL error and warning messages that may be displayed as you use ISQL.

2

Getting Started with ISQL

This chapter examines how and when ISQL commands are used. The following topics are discussed:

- Invoking ISQL.
- Starting a DBE Session.
- Leaving ISQL.
- Managing Transactions.
- Using SQL and ISQL Commands.
- Using ISQL Files and Buffers.
- Getting Help.
- Setting Command Options.
- Using Other Programs within ISQL.
- Handling Errors.

Invoking ISQL

You invoke ISQL by issuing the following command:

    $ isql

ISQL displays a banner followed by an input prompt.

```
    IIIIIIII    SSSSSSSS    QQQQQQQ       LL
       II       SS         QQ    QQ       LL
       II       SS         QQ    QQ       LL
       II        SSSSSS    QQ    QQ       LL
       II             SS   QQ    QQ       LL
       II             SS   QQ    QQ       LL
    IIIIIIII    SSSSSSS     QQQQQQ QQ     LLLLLLLLL


       MON, JAN 13, 1992,  9:55 AM
    HP36217-02A.F0.00          Interactive SQL/9000          ALLBASE/SQL
    (C)COPYRIGHT HEWLETT-PACKARD CO. 1982,1983,1984,1985,1986,1987,1988,
    1989,1990,1991,1992. ALL RIGHTS RESERVED.


    isql=>
```

ISQL interacts with several files each time it is invoked:

- isqlpro—the profile file.
- isqlsyn—the synonym file.
- hpsqlcat—the message file.
- isqlwel—the welcome file.

The profile and synonym files are discussed later in this chapter under "Using ISQL Files and Buffers." The message file contains ISQL and other ALLBASE/SQL messages.

The welcome file contains the ISQL banner and prompt to be displayed when ISQL is invoked, as shown above. The ISQL prompt is defined on the the first line of the welcome file, although you can override this prompt with a SET PROMPT command submitted either from a profile file or interactively.

Starting a DBE Session

Before you can execute ISQL or SQL commands that operate on a DBEnvironment, you must begin a DBE session. You begin a DBE session with either the CONNECT statement, as illustrated below, or the START DBE statement:

```
isql=> CONNECT TO '../sampledb/PartsDBE';
```

Refer to the *ALLBASE/SQL Reference Manual* for additional information on starting DBE sessions. Appendix C of that manual describes how to set up a copy of the sample DBEnvironment PartsDBE for practice use.

Leaving ISQL

To terminate an ISQL session, you use the END or the EXIT command as follows:

```
isql=> END;

$
```

If a DBEnvironment transaction is in progress when you submit the END or EXIT command, ISQL prompts you as follows:

```
isql=> END;
A transaction is in progress.  Commit work (Y/N)>
```

ISQL automatically terminates the transaction based on your response.

If you enter YES, ISQL issues a COMMIT WORK statement. When the COMMIT WORK statement is executed, changes to the database are made permanent. If you enter NO, ISQL issues a ROLLBACK WORK statement. When the ROLLBACK WORK statement is executed, changes to the database that have not been committed are undone. ISQL then automatically terminates your DBE session by processing a RELEASE statement.

Managing Transactions

A **transaction** consists of one or more SQL statements that are grouped together to form a unit of work. For example, if you wanted to transfer money from a savings to a checking account, the withdrawal and the deposit would both occur within the same transaction. A transaction begins with a BEGIN WORK statement and ends with either a COMMIT WORK or a ROLLBACK WORK statement. Either all the statements or none of the statements are executed.

How ISQL Manages Transactions

ISQL automatically processes a BEGIN WORK statement whenever you successfully submit *most* SQL statements and a transaction is not already in progress:

```
isql=> CONNECT TO '../sampledb/PartsDBE';
isql=> UPDATE STATISTICS FOR TABLE PurchDB.Parts;
isql=> BEGIN WORK;
Transaction already started.  (DBERR 2103)
isql=>
```

In this example, the UPDATE STATISTICS statement automatically does an implicit BEGIN WORK. Thus the explicit BEGIN WORK creates an error condition.

The following SQL statements do *not* cause ISQL to process a BEGIN WORK statement:

```
BEGIN ARCHIVE
BEGIN WORK
CHECKPOINT
COMMIT ARCHIVE
CONNECT
RELEASE
START DBE
STOP DBE
TERMINATE USER
```

ISQL also automatically processes a BEGIN WORK statement whenever you successfully submit the following ISQL commands and a transaction is not already in progress:

```
INFO
INPUT
INSTALL
LOAD
UNLOAD
```

Using SQL SAVEPOINT and ROLLBACK WORK Statements

Within a transaction, you can set **savepoints**. Work accomplished after a savepoint can be undone at any time prior to the end of the transaction. When you issue the SAVEPOINT statement in ISQL, you are assigned a savepoint number, starting at 1 with each new transaction. You reference the savepoint number in a ROLLBACK WORK statement to undo work done since the referenced savepoint was established. For example:

```
isql=> SAVEPOINT;
Savepoint number is 1
Use this number to do ROLLBACK WORK TO 1.
isql=> Command; Command ... ;
isql=> ROLLBACK WORK TO 1;
isql=>
```

ISQL automatically terminates transactions for you in several instances:

- ISQL issues a COMMIT WORK statement when the SET AUTOCOMMIT option is ON and you are using the ISQL command INPUT, INSTALL, or LOAD.

- ISQL issues a COMMIT WORK or ROLLBACK WORK statement when you respond to the exit prompt described earlier in this chapter under "Leaving ISQL."

When accessing multiuser DBEnvironments from ISQL, you may need to submit the COMMIT WORK or ROLLBACK WORK statement frequently to improve concurrency.

More information on managing SQL transactions is provided in the *ALLBASE/SQL Database Administration Guide*.

Using SQL and ISQL Commands

When you are in ISQL, you can issue both ISQL and SQL commands. However, you can use ISQL commands only when you are in ISQL. SQL statements can be used in application programs also.

You can also invoke other programs from ISQL and issue commands from them. For more information, refer to "Using Other Programs within ISQL" in this chapter.

Terminating Commands

All commands entered through ISQL must be terminated with a semicolon. When accepting commands, ISQL does not process anything you enter until it detects a semicolon followed by a carriage return. Notice in the example below that, when accepting commands from a terminal and a semicolon is not the last character on the line, ISQL displays a line-continuation prompt (>) whenever you press (Return):

```
isql=> UPDATE STATISTICS FOR TABLE PurchDB.Parts; UPDATE
> STATISTICS FOR TABLE PurchDB.SupplyPrice; UPDATE STATISTICS FOR
> TABLE PurchDB.Orders
> ;
isql=>
```

A command can span multiple lines. Conversely, you can submit multiple commands on a single line.

When your input to ISQL is *only a single character* and you are executing commands interactively or from a command file, you can omit the semicolon, as follows:

```
A transaction is in progress. Commit work  (Y/N)> N
END OF PROGRAM
```

Prompting Mode

ISQL provides a special mode of command entry called **prompting mode**. Prompting mode is available to you when you submit ISQL commands from a terminal. In prompting mode, you can enter part of an ISQL command, and ISQL prompts you for subsequent options and parameters:

```
isql=> LIST;

Option (s[et], f[ile], i[nstall], or h[istory])> File;
File name> MyFile;
```

The contents of the named file are displayed.

Terminate both your original input and your response to each prompt with a semicolon and ⌷Return⌷.

In prompting mode, ISQL skips prompts when you provide more input at one time:

```
isql=> LIST;

Option (s[et], f[ile], i[nstall], or h[istory])> FILE MyFile;
```

The contents of the named file are displayed.

```
isql=> LIST FILE;

File name> MyFile;
```

The contents of the named file are displayed.

ISQL accepts responses to its prompts either in the current language or in n-computer. Refer to the section on "Native Language Support" in the "Introduction" for an explanation of how to set the current language.

Aborting Commands

Respond to an ISQL prompt with a slash (/) or two slashes (//) anytime you want to abort the command you are currently entering. For example:

```
isql=> ERASE;
File name> /
isql=> SELECT * FROM
> //
isql=> EXIT;
A transaction is in progress.  Commit work (Y/N)> /
isql=>
```

Special Characters in Commands

Table 2-1 lists characters that have special meaning to ISQL.

Table 2-1. Special Characters in ISQL

Special Character	Meaning to ISQL
'	Delimiter
"	Delimiter
;	Command terminator
&	Parameter indicator
--	Comment marker
/* ... */	Comment marker
\	Escape character (initially set to \, but modifiable with the SET command)

When you enter SQL statements with embedded quotation marks, you need to double all embedded quotation marks so that ISQL does not interpret the embedded quotation marks as closing delimiters. For example:

```
isql=> UPDATE PurchDB.Vendors SET VendorRemarks =
> 'Don''t use this vendor unless it''s an emergency!'
> WHERE VendorNumber = 9006;
```

Object names included in double quotes cannot be split across lines.

You can insert comments into command files in the following ways:

- Using two hyphens (--).
- Using slashes and asterisks (/* and */).

When using the hyphens, you prefix the comment with two hyphens on each line. The comment terminates at the end of the current line.

When using the slash and asterisk, enter /* before the comment starts and */ at the end of the comment. In this case, comments can span lines; you only need to put the */ at the end of the comment.

The following example shows a command file using both types of comment markers:

```
/* The following statement is used when we have a rush
order for Graphics Monitor Adaptors.*/

SELECT * FROM PurchDB.SupplyPrice    --This statement selects values
 WHERE PartNumber = '1723-AD-01'     --from the SupplyPrice table.
 AND DeliveryDays <30;
```

In an ISQL command, when you want special characters to have a literal meaning to ISQL, precede the character with the escape character currently defined in the ESCAPE option of the SET command. The escape character is initially set to a backslash (\).

In the following example, the ISQL CHANGE command is used to edit a command placed in the command buffer with the HOLD command. The escape character (\) is needed so that ISQL interprets the single quotation marks being inserted as quotation marks, not delimiters. The slashes (/) are delimiters for the CHANGE command.

```
isql=> HOLD SELECT * FROM PurchDB.Parts WHERE PartNumber<1300;
isql=> CHANGE /1300/\'1300\'/;

SELECT * FROM PurchDB.Parts WHERE PartNumber<'1300';

isql=>
```

In the following example, the ISQL system interrupt character (!) is used to submit *cat* and a *date* commands. The escape character is needed so that ISQL does not interpret the first semicolon as a command terminator.

```
isql=> !cat isqlpro\;date;
```

If any of the special characters in Table 2-1 is replaced by another character in the character set of a native language you are using, you must type the replacement character instead. For example, in the Japanese Kana 8 character set, the ASCII "\" is replaced by the yen symbol. Therefore, ISQL users whose current language is Katakana or Japanese must type a yen symbol wherever ISQL syntax requires a "\".

Parameters in Commands

You can include parameters in your commands by using an ampersand (&) and a digit from 1 through 100. Using a parameter lets you assign different values to parts of a command each time you execute the command. For example, you might use a parameter for a table name or a column name. When you submit the command for execution, ISQL prompts you for parameter values, as follows:

```
isql=> SELECT &1 FROM &2;
SELECT &1 FROM &2;
Value for parameter &1> *;
Value for parameter &2> PurchDB.Parts;
```

   Query result is displayed.

Parameters are convenient to use in commands submitted from files or from the command buffer. Command files and the command buffer are discussed in the section "Using ISQL Files and Buffers" below. As in the following example, when you use the START command to execute the commands in the file or buffers, ISQL prompts you for parameter values:

```
isql=> START MyFile;
UPDATE STATISTICS FOR TABLE &1;
Value for parameter &1> PurchDB.Parts;
```

   Command file processing continues.

Alternatively, you can submit parameter values in the START command itself as shown here:

```
isql=> START MyFile (PurchDB.Parts);
UPDATE STATISTICS FOR TABLE &1;
```

Command file processing continues.

Within any single command file or command buffer, each parameter number assumes one value. This feature lets you use the same parameter more than once, but you can supply its value only once. For example:

```
isql=> SELECT &1 FROM &2 WHERE &1 = &3;
SELECT &1 FROM &2 WHERE &1 = &3;
Value for parameter &1> SalesPrice;
Value for parameter &2> PurchDB.Parts;
Value for parameter &3> 1000;
```

Query result is displayed.

Using ISQL Files and Buffers

ISQL has a **command router** that determines whether a command is an SQL statement or an ISQL command and then routes it to SQLCore for execution. The ISQL commands that entail database access, such as data loading operations, are converted into SQL statements by the ISQL command manager. The command router also looks up synonyms when a synonym file is used.

As Figure 2-1 illustrates, ISQL's command router that accepts commands from the following sources:

- The terminal.
- Files containing commands.
- The command buffer.
- The command history buffer.
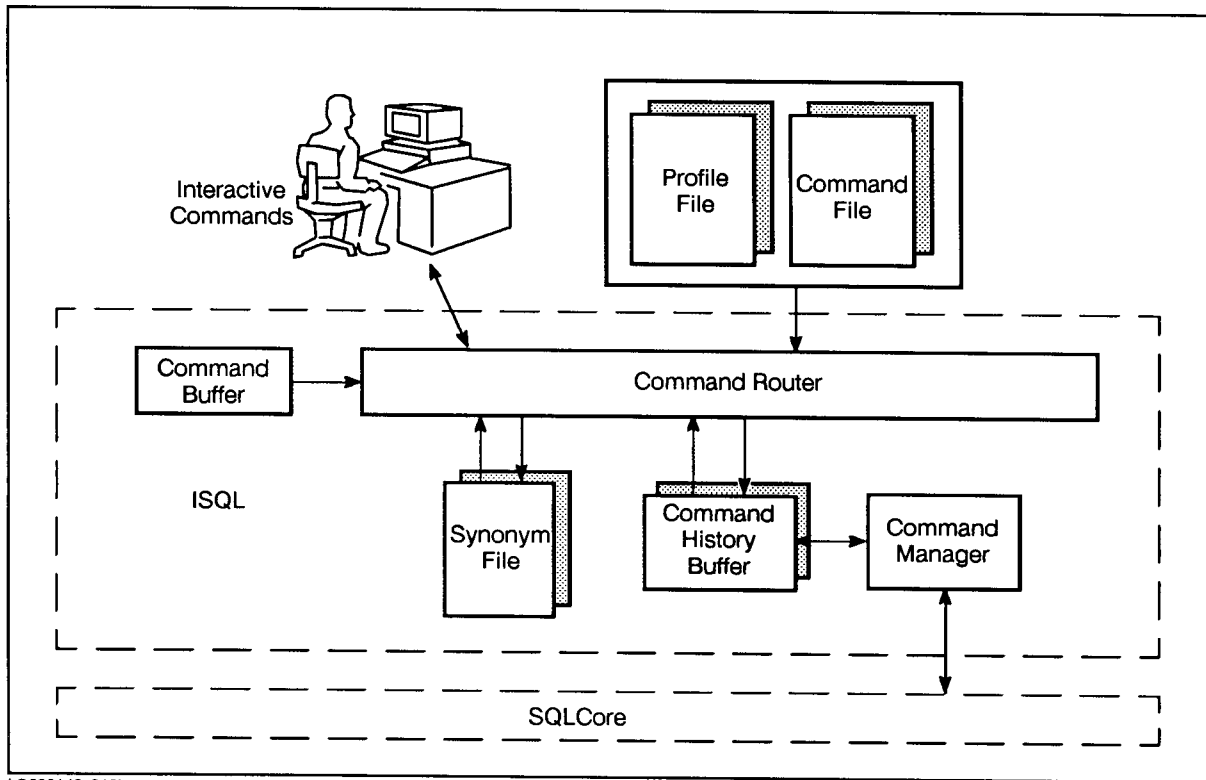
LG200142_010b

Figure 2-1. Command Sources

Entering Interactive Commands from the Terminal

You submit one or more commands when you see the input prompt. ISQL processes the commands, then displays the input prompt again.

Files Containing Commands

A **command file** is a file that contains one or more commands you want to execute using ISQL. Command files are useful when you want to enter lengthy or repetitive commands. Any SQL statement you can enter from a terminal can be put in a command file, and any ISQL command except REDO can be put in a command file. Commands in a command file can contain parameters. To execute commands in a command file, you by issue the START command. Control returns to the terminal when ISQL stops processing commands from the file.

To maintain command files, you can use the following ISQL commands:

■ EDIT invokes that editor named in the EDITOR option of the SET command.
■ STORE saves the current contents of the command buffer in a file.
■ ERASE purges files.
■ LIST FILE displays files at the terminal.
■ RENAME changes the names of files.
■ RECALL FILE places up to 32K bytes of a command file into the command buffer.
■ START executes the commands in the command file.

Figure 2-2 summarizes the ISQL commands related to command file creation, maintenance, and use. Refer to "Command Buffer" and "Command History Buffer" later in this chapter for an explanation of the buffer areas.
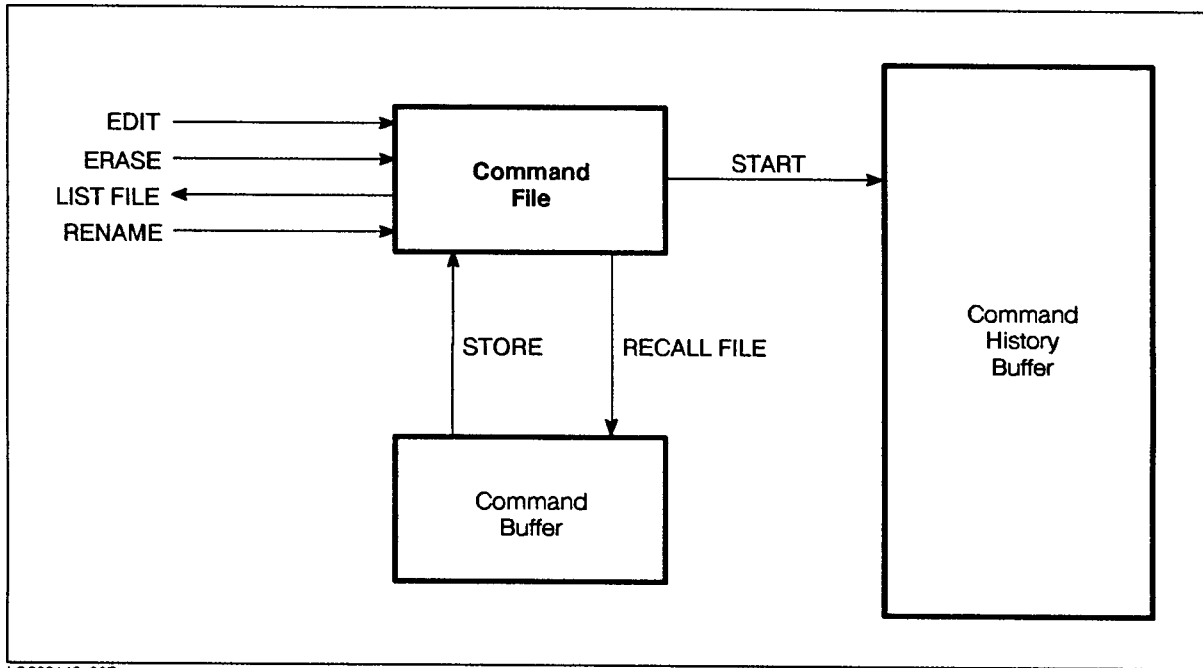
LG200142_007

Figure 2-2. ISQL Commands Related to Command Files

You can create command files with the editor on your system. For example:

```
isql=> EDIT LoadDB;
/* Command file to load PartsDBE tables            */
/* Command to load table PurchDB.Parts             */
LOAD FROM EXTERNAL PartData TO PurchDB.Parts
PartNumber              1     16
PartName               17     30    ?
SalesPrice             47     10    ?
END  NO;
COMMIT WORK;
SELECT * FROM PurchDB.Parts;
PAUSE;
   .
   .
   .
:wq
isql=>
```

You can format commands however you want to in command files, but you must terminate each ISQL and SQL statement in the file with a semicolon. You can also insert comments into the command file; enter a /* before the comment starts and a */ at the end of the comment. Comments can span lines, and in this case you only need to put the */ at the end of the comment. Do not put information after column 80 in command files if you want ISQL to process it, because ISQL reads only up to 80 bytes per line when executing commands from a command file.

You execute command files by using the START command as follows:

```
isql=> START LoadDB;
```

You can include additional START commands in a command file. The maximum number of STARTs permitted (including the first START and all levels of nesting) is 10.

During execution of command files, what you see at the terminal depends on the setting of the ECHO option of the SET command and whether you have included the PAUSE option:

■ When ECHO is ON, ISQL displays lines at the terminal as they are read from the command file. Both comments and commands in the file are displayed. Regardless of the setting of this option, however, normal command output and error messages are displayed at the terminal.

■ When you specify the PAUSE option in your command file following a SELECT statement, ISQL temporarily suspends command file processing to let you browse through and optionally print the query result at your terminal. The discussion under "SELECTSTATEMENT" in chapter 4 shows how to browse through displayed query results. When you enter END; at the terminal, ISQL resumes reading lines from the command file.

If you do not specify the PAUSE option following a SELECT statement in a command file, ISQL displays the first group of data rows of the query result, then resumes processing commands from the file; in this case, you must include END; in your command file to terminate the query.

If ISQL encounters an error while processing commands from a command file, it terminates command file processing if the CONTINUE option value of the SET command is OFF. If CONTINUE is ON, ISQL continues processing any subsequent commands in the file.

Creating a Profile File

The **profile file** is a special kind of command file. When you invoke ISQL, the commands in your profile file are executed before the input prompt appears. Your profile file must reside in your current working directory and be named *isqlpro*.

You can create a profile file with an editor on your system, or with ISQL STORE or EDIT commands as follows:

```
isql=> EDIT;

Editor is invoked.
 ...
SET ECHO ON;
CONNECT TO 'mydbe';
:wq isqlpro
isql=>
```

This profile file contains one ISQL and one SQL statement. The ISQL command sets the SET ECHO option ON. The SQL statement establishes a DBE session.

Running ISQL From a Command File

You can run ISQL from a command file by specifying the file name as part of the ISQL command invocation. Example:

    $ isql < *CommandFile*

If you specify a command file and if isqlpro also exists, the commands in isqlpro are executed first.

Running ISQL in the Background

You can run ISQL in the background by using a command file as redirected input to ISQL. As an example, imagine a command file that could be used to create the sample DBEnvironment PartsDBE. The command file, which might be called *creajob*, starts several other ISQL command files (located in the *sampledb* directory) to define the sample DBEnvironment. Here is the content of *creajob*:

```
/* This file starts the various ISQL command files that create
      and load the PartsDBE DBEnvironment. */

SET ECHO ON;
START sampledb/STARTDBE;
!echo The DBEnvironment PartsDBE is now created!;
START sampledb/CREATABS;
!echo The Tables and Views are now created!;
COMMIT WORK;
START sampledb/LOADTABS;
!echo The Tables and Views are now loaded and statistics updated!;
COMMIT WORK;
START sampledb/CREAINDX;
!echo The Indexes are now created!;
COMMIT WORK;
START sampledb/CREASEC;
!echo The Security schema for PartsDBE is now created!;
COMMIT WORK;
EXIT;
!echo Creation and Loading of PartsDBE is now complete!;
```

The ISQL command file named STARTDBE configures the DBEnvironment. Tables and other objects are defined in the command file named CREATABS. The command file named LOADTABS contains ISQL LOAD commands for loading the tables. The command file named CREAINDX contains SQL statements for creating indexes. The command file named CREASEC contains the SQL statements that create authorization groups and grant users and groups specific authorities.

You run *creajob* in the background as follows:

    $ isql < creajob > isqlout &

Refer to the "Sample DBEnvironment" appendix in the *ALLBASE/SQL Reference Manual* for a complete description of the ISQL command files used to create PartsDBE.

When executing ISQL in the background, the output file created does not contain an echo of the commands executed in ISQL. It does contain visible escape characters in query results.

The Command Buffer

The command buffer is a 32K byte area for storing one or more commands for the duration of an ISQL session. A single command greater than 32K bytes is truncated and is not executed. You put commands into the command buffer by using the ISQL RECALL or HOLD command. You execute commands in the command buffer by using the ISQL START command. Figure 2-3 summarizes the ISQL commands that operate on the command buffer.

LG200142_008
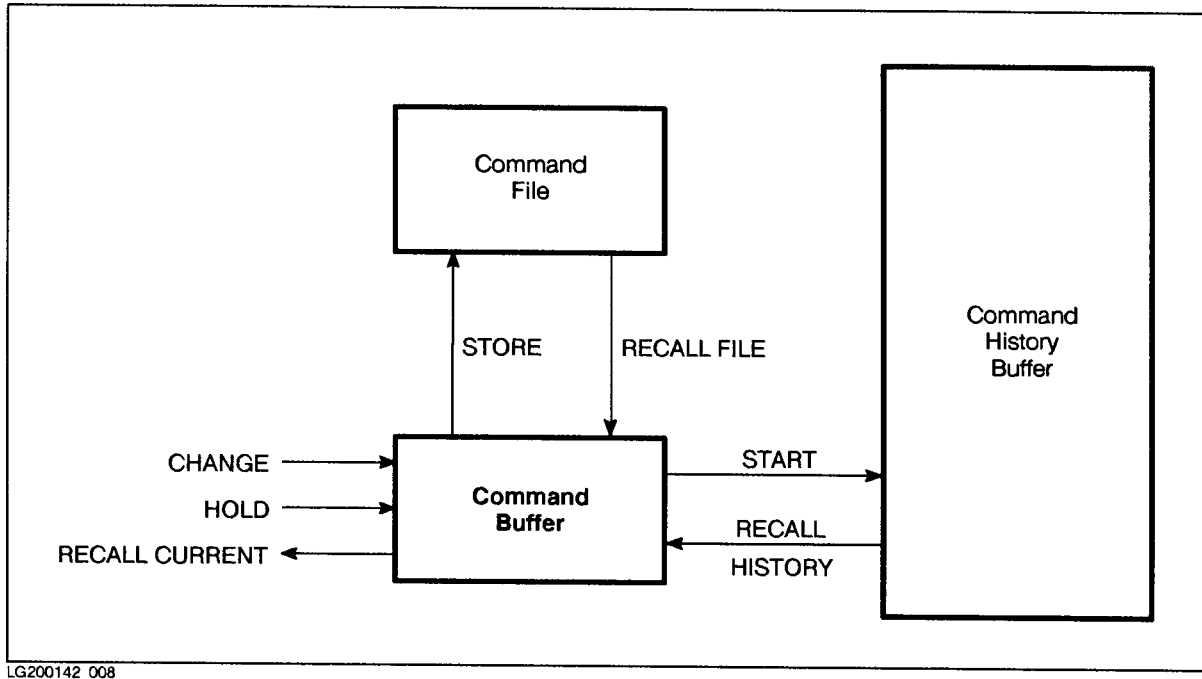
Figure 2-3. ISQL Commands Related to the Command Buffer

When you first invoke ISQL, the command buffer is empty. You can use one of the following commands to put information into the command buffer:

- HOLD puts one or more commands you key in at the terminal into the command buffer. To put multiple commands into the command buffer with the HOLD command, type the current escape character before the semicolon separating commands as shown below:

      isql=> HOLD UPDATE STATISTICS
      > FOR TABLE PurchDB.Parts\;INFO PurchDB.Vendors;

- RECALL FILE puts the contents of a file into the command buffer and displays it.

- RECALL HISTORY puts a command from the current command history buffer into the command buffer and displays it. The command history buffer is discussed in the following section.

You can use the RECALL CURRENT command to identify the commands currently in the buffer. For example:

      isql=> RECALL CURRENT;

      UPDATE STATISTICS FOR TABLE PurchDB.Parts;INFO PurchDB.Vendors;

You can change, store, or execute the contents of the command buffer with the CHANGE, STORE, and START commands, respectively:

- CHANGE replaces one string with another string, which can be null. You may replace one or all occurrences of the string in the command buffer.

- STORE keeps the contents of the command buffer in a file.

- START executes the command(s) in the command buffer. The commands may contain parameters.

The Command History Buffer

When you invoke ISQL, the command history buffer is empty. As you submit commands, ISQL puts the first 32K bytes of each command into this buffer. As illustrated in Figure 2-4, ISQL and SQL commands submitted from any source are put into the command history buffer. The RECALL HISTORY and LIST HISTORY commands are the only commands not placed in this buffer.
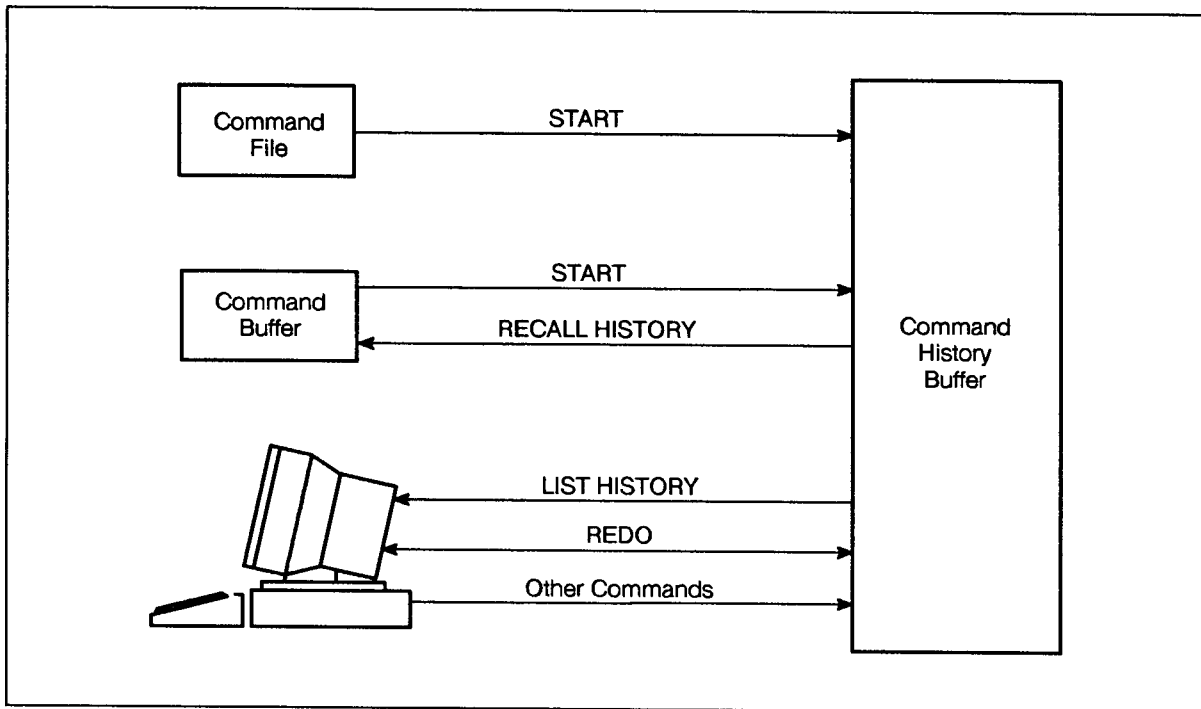
LG200142_009

Figure 2-4. ISQL Commands Related to the Command History Buffer

The command history buffer holds the 10 most recently submitted commands. Commands in the command history buffer are numbered from 1 through 10, with 1 being the most recently submitted command. When you submit multiple commands at the same time, each command occupies a separate slot in the history buffer.

The following three commands operate on the commands currently in the command history buffer:

- LIST HISTORY displays one or all of the commands in the command history buffer; you
  can use an @ to indicate that you want to see the entire command history buffer. For
  example:

```
isql=> LIST HISTORY @;

1      SELECT * FROM PurchDB.Parts;
2      SELECT * FROM PurchDB.Vendors;
3      SELECT * FROM PurchDB.PartInfo;
4      CONNECT TO 'sampledb/PartsDBE';
5
6
7
8
9
10

isql=>
```

- RECALL HISTORY puts one command from the command history buffer into the
  command buffer and displays it. For example:

```
isql=> RECALL HISTORY 2;

SELECT * FROM PurchDB.Vendors;

isql=>
```

- REDO lets you edit any of the commands in the command history buffer, then execute the
  edited command if you desire. As the following example illustrates, you use the REDO I
  subcommand to insert data and the X subcommand to execute the edited command:

```
isql=> REDO 2;
SELECT * FROM PurchDB.Vendors;
          Itatistics
SELECT * FROM PurchDB.Vendorstatistics;
X
```

  SELECT statement is executed.

Refer to the REDO command in chapter 4, "ISQL Commands," for a complete explanation
of this command's editing features.

When you submit an ISQL command in prompting mode, as in the following example, only
the part of the command first submitted is stored in the command history buffer:

```
isql=> LOAD FROM INTERNAL;

Input file name> InFile;
'InFile' cannot be found.  (DBERR 17)
isql=> RECALL HISTORY 1;

LOAD FROM INTERNAL;
isql=> START;

Input file name> IntFile;
Table name> PurchDB.Vendors;
Number of rows processed is 15
COMMIT WORK to save to DBEnvironment

isql=>
```

Using a Synonym File

You can equate ISQL and SQL commands to your own command verbs by using a **synonym
file**. For example, you can define UNDO to be the synonym for ROLLBACK WORK. When
you enter UNDO, ISQL issues the SQL ROLLBACK WORK statement. A synonym file must
be named *isqlsyn* and must exist in your current working directory when you invoke ISQL.

Each line in a synonym file should contain an ISQL or SQL command verb, followed by
at least one blank, then a synonym. There can be as many as 100 synonyms per file. The
ISQL or SQL command can contain as many as 60 bytes, including embedded ASCII blanks.
Synonyms can be as long as 16 bytes, including embedded ASCII blanks. Multiple-word verbs
and synonyms should be delimited by ASCII double quotation marks as shown below:

```
isql=> EDIT;
```

Editor is invoked.
```
 ...
"COMMIT WORK"  CW
REDO           FIX
"CONNECT TO '../sampledb/PartsDBE'"   OPEN
:wq isqlsyn
isql=>
```

To invoke a synonym that contains embedded blanks, enclose it in double quotation marks. For example, if a CONNECT statement is defined in your synonym file as "OPEN DB", invoke it as follows:

```
isql=> "OPEN DB";
```

Getting Help

ISQL has a HELP command you can use to obtain information about the function and syntax of any ISQL command or SQL statement. The following example obtains information about the DELETE statement:

```
isql=> HELP DELETE;


SCOPE:  ISQL or Application Programs

The DELETE statement deletes a row or rows from a table.

==========
  SYNTAX
==========

DELETE FROM {[Owner.]TableName} [WHERE SearchCondition]
      {[Owner.]ViewName }

==========
  EXAMPLE
==========

DELETE FROM PurchDB.Orders
    WHERE OrderDate < '19830701' ;


isql=>
```

When you specify a verb that is used in more than one command, ISQL lists all the commands that qualify and prompts you to enter the complete command. If you do not specify Description, Syntax, or Example in the command line, all three types of help text are displayed.

Setting Command Options

Various options are controlled with the SET command.

The SET options have initial settings, which you can change for the duration of an ISQL session with the SET command. To determine the current value of a SET option, you use the LIST SET command. The following example illustrates the initial values of all SET options:

```
isql=> LIST SET @;

autoc[ommit]      - OFF
autol[ock]        - OFF
autos[ave]        - 5
c[ontinue]        - ON
conv[ert]         - OFF
ec[ho]            - OFF
echo_[all]        - OFF
ed[itor]          - /usr/bin/vi
es[cape]          - \
exit[_on_dberr]   - OFF
exit_on_dbwarn    - OFF
fl[agger]         -
f[raction]        - 2
load_b[uffer]     - 16384
load_e[cho]       - ON
n[ull]            -
ou[tput]          - isqlout
ow[ner]           -
pa[gewidth]       - 32767
pr[ompt]          - isql=>

isql=> SET ECHO ON;
isql=>
```

The @ is an ISQL wild-card character meaning *all*. If your system line-kill character is set to @, precede the @ with the current HP-UX escape character. For example:

```
isql=> LIST SET \@;
```

As indicated in Table 2-2, most SET options control ISQL processing related to specific commands. Note that except the SELECT statement, all commands under "Related Command" are ISQL commands.

Table 2-2. SET Command Options

Category	Related Command	SET Option	Effect

Data loading; application program support	`INPUT`
	`LOAD`
	`INSTALL`
		`AUTOCOMMIT`	Determines whether ISQL issues COMMIT WORK automatically.

	`LOAD`	`AUTOLOCK`	Determines whether tables are locked when loaded.
	`INPUT` `LOAD`	`AUTOSAVE`	Determines how often automatic COMMIT WORK is performed.
	`LOAD`	`CONVERT`	Determines whether data will be converted during the load.
	`LOAD` `UNLOAD`	`LOAD_BUFFER`	Determines the size of the load/unload buffer.
	`LOAD` `UNLOAD`	`LOAD_ECHO`	Determines when status messages are displayed.

Table 2-2. SET Command Options (continued)

Category	Related Command	SET Option	Effect
Query results	`SELECT`	`FRACTION`	Determines how many decimal digits are displayed.
		`NULL`	Determines the character displayed when a value is null.
		`OUTPUT`	Determines the file to which the query result is sent.
		`PAGEWIDTH`	Determines the maximum row length in the query result.
Command files	`START`	`CONTINUE`	Determines whether processing is to be continued if an error is encountered.
		`ECHO`	Determines whether lines in the file are displayed as they are read.
Editing	`EDIT`	`EDITOR`	Determines the editor invoked.
ISQL prompt		`PROMPT`	Determines the input prompt string.
Special characters		`ESCAPE`	Determines the escape character.
Batch processing		`ECHO_ALL`	Determines whether user input is displayed.
		`EXIT_ON_DBERR`	Determines whether ISQL terminates immediately when an error is encountered.
		`EXIT_ON_DBWARN`	Determines whether ISQL terminates immediately when a warning is encountered.

Using Other Programs within ISQL

Using the System Editor

You can invoke any editor on your system from ISQL. Use the ISQL SET command with the EDITOR option to define the editor you want to use. For example:

```
isql=> SET EDITOR emacs;
isql=> EDIT;
```

For complete information on using the SET command, refer to chapter 4, "ISQL Commands."

Using System Commands and Programs

You can execute operating system commands by using the SYSTEM command.

You can specify the SYSTEM command or the system interrupt character (!) before an operating system command and return immediately to ISQL as follows:

```
isql=> SYSTEM date;
Tues Aug 25  16:29:18 PDT 1987
isql=>
```

You can also temporarily leave the ISQL environment to execute operating system commands, then return to ISQL by entering EXIT. For example:

```
isql=> !
$ date

Mon Aug 24  16:28:45 PDT 1987

$ ps
  PID    TTY     TIME    COMMAND
11614   ttyp2   0:00    sh
11593   ttyp2   0:00    jupmars
11592   ttyp2   0:01    isql
10767   ttyp2   0:00    csh
11615   ttyp2   0:00    sh
11632   ttyp2   0:00    ps

$ exit
isql=>
```

Using SQLUtil

SQLUtil is a program that lets you perform maintenance tasks on a DBEnvironment. SQLUtil commands cannot be issued directly from ISQL. You must first invoke SQLUtil from ISQL by using the SQLUTIL command, as shown here:

```
isql=> SQLUTIL;
```

The SQLUtil banner and prompt (>>) are displayed. Then you can execute SQLUtil commands until you enter an EXIT or QUIT command. For complete information on using

SQLUtil, refer to the "DBA Tasks and Tools" chapter of the *ALLBASE/SQL Database Administration Guide*.

Using SQLGEN

SQLGEN is a program that allows you to re-create a given DBEnvironment.

SQLGEN commands cannot be issued directly from ISQL. You must first invoke SQLGEN from ISQL by using the SQLGEN command as follows:

```
isql=> SQLGEN;
```

The SQLGEN banner and prompt (>>) are displayed. You can then execute SQLGEN commands until you enter an EXIT or QUIT command.

For complete information on using SQLGEN, refer to the "DBA Tasks and Tools" chapter of the *ALLBASE/SQL Database Administration Guide*.

Handling Errors

Several types of errors can occur while you are using ISQL:

- Command errors.
- DBEnvironment errors.
- Unexpected ISQL or DBEnvironment termination.

Usually, you see an error message when an error occurs. Error messages are taken from the message catalog. The name of the default catalog is /usr/lib/nls/n-computer/hpsqlcat. If you are a native language user, the catalog's name is /usr/lib/nls/$LANG/hpsqlcat, where $LANG is the current language. If this catalog is not available, ALLBASE/SQL uses the default catalog instead. Refer to the *ALLBASE/SQL Message Manual* for information on interpreting error and warning messages and suggestions on correcting exception conditions. You can also use the HELP command to obtain information on command use and structure if a command error occurs.

Command Errors

Commands may contain either syntax or semantic errors. **Syntax errors** involve erroneous command structure, as when a required parenthesis or keyword is omitted. **Semantic errors** involve erroneous information, as when the name of a table is misspelled.

When you submit an ISQL or SQL command from the terminal and it contains errors, you can recall it from the command history buffer, correct it, then resubmit it. For example:

```
isql=> UPDATE STATSTICS FOR TABLE PurchDB.Parts;

UPDATE STATSTICS FOR TABLE PurchDB.Parts;
        |
Unexpected keyword.  (DBERR 1006)
isql=> RECALL HISTORY 1;

UPDATE STATSTICS FOR TABLE PurchDB.Parts;
```

```
isql=> CHANGE /STATS/STATIS/;

UPDATE STATISTICS FOR TABLE PurchDB.Parts;

isql=> START;
isql=>
```

When you are submitting commands from a command file, the CONTINUE option of the SET command determines what ISQL does when it encounters an erroneous command. If CONTINUE is ON, ISQL continues processing any subsequent commands in the command file. If CONTINUE is OFF, ISQL discontinues processing commands from the file and displays the input prompt.

When you submit commands from the command buffer, ISQL continues processing any subsequent commands in the buffer when it encounters an erroneous command.

When the EXIT_ON_DBERR option of the SET command is ON, ISQL will terminate immediately if an SQL error occurs. The EXIT_ON_DBERR option has precedence over the CONTINUE option.

DBEnvironment Errors

Some errors may be caused by the following conditions in the DBEnvironment:

- A DBEnvironment is not started yet.
- Resources are insufficient.
- A deadlock has occurred.

Refer to the *ALLBASE/SQL Database Administration Guide* information on handling these kinds of errors.

Unexpected ISQL or DBEnvironment Termination

If ISQL stops running or a STOP DBE command is issued while you are submitting SQL statements, work done by any transactions in progress at termination is automatically undone when the DBEnvironment is next started up. Only work up to the last COMMIT WORK executed prior to termination permanently affects the DBEnvironment.

Checking the Exit Status

When running ISQL in a script, you can check the $? exit status to ensure that ISQL completed successfully. ISQL sets the exit status to the number of SQL errors encountered. If an SQL error is not encountered, ISQL sets the exit status to 0. The exit status may contain a value in the range 0 - 255.

3

Using ISQL for Database Tasks

This chapter explains how to use ISQL to change and display data in the DBEnvironment. The following topics are discussed:

- Creating Objects.
- Controlling Access to Objects.
- Manipulating Data.
- Unloading and Loading Tables.
- Using Modules.

Creating Objects

You can use ISQL to create database objects such as tables, views, and indexes. To do this, you use the SQL CREATE statements, as in the following example:

```
isql=> CREATE TABLE MyTable (col1 smallint, col2 char(10)) in MyFS;
```

A table named MyTable is created with two columns: The first column can contain integer data and the second character data. The table is stored in MyFS.

Statements used to create and redefine objects in the database are collectively referred to as **data definition language** (DDL) statements. Refer to the "SQL Statements" chapter of the *ALLBASE/SQL Reference Manual* for complete information on using the following SQL CREATE statements:

```
CREATE DBEFILE
CREATE DBEFILESET
CREATE TABLE
CREATE VIEW
CREATE INDEX
CREATE GROUP
CREATE TEMPSPACE
```

You can also remove objects from a DBEnvironment through ISQL using the following SQL DROP statements:

```
DROP DBEFILE
DROP DBEFILESET
DROP TABLE
DROP VIEW
DROP INDEX
DROP GROUP
DROP MODULE
DROP TEMPSPACE
```

Refer to the "SQL Statements" chapter of the *ALLBASE/SQL Reference Manual* for complete information on using these DROP statements.

For complete information on configuring a new DBEnvironment, refer to the chapters on "DBEnvironment Configuration and Security" and "Database Creation and Security" in the *ALLBASE/SQL Database Administration Guide*.

Controlling Access to Objects

You can use ISQL to grant or revoke various privileges on data objects as follows:

```
isql=> GRANT SELECT ON MyTable to User1;
```

User1 is granted authority to retrieve data on MyTable.

Statements used to control access to the DBEnvironment are collectively referred to as **data control language** (DCL) statements. Refer to the "SQL Statements" chapter of the *ALLBASE/SQL Reference Manual* for complete information on using the following SQL data control statements:

```
GRANT
REVOKE
CREATE GROUP
DROP GROUP
ADD TO GROUP
REMOVE FROM GROUP
TRANSFER OWNERSHIP
```

Manipulating Data

You can use ISQL to manipulate and access the data in your database by using the following SQL and ISQL commands:

INSERT	*An SQL statement.*
INPUT	*An ISQL command.*
SELECT	*An SQL statement.*
DELETE	*An SQL statement.*
UPDATE	*An SQL statement.*
INFO	*An ISQL command.*

The SQL statements in this list are collectively referred to as **data manipulation language** (DML) statements. INPUT and INFO are ISQL commands which support the data manipulation function.

A brief description of each command is presented in the next sections. Refer to the *ALLBASE/SQL Reference Manual* for complete information on using the SQL statements. Refer to the "ISQL Commands" chapter of this manual for complete information on using INPUT and INFO.

Using the INSERT Statement

The basic SQL statement for entering data into a table is the INSERT statement, as shown in the following example:

```
isql=> INSERT INTO PurchDB.Parts VALUES
> ('2010-S-01','Synthesizer, monophonic',499.00);
isql=>
```

A single row is inserted into the Parts table.

Using the INPUT Command

As an alternative to the SQL INSERT statement, you can use the ISQL INPUT command shown here.

```
isql=> INPUT;

Table name> PurchDB.Parts;
Column name> (PartNumber, PartName);
1> ('2010-S-01','Synthesizer, monophonic');
2> ('2010-S-02','Synthesizer, stereophonic');
3> ('2010-S-03',null);
4> END;
Number of rows processed is 3
COMMIT WORK to save to DBEnvironment.

isql=>
```

ISQL provides the following options for committing rows during INPUT command execution:

■ When the AUTOCOMMIT option of the SET command is ON, ISQL automatically processes a COMMIT WORK statement every time the number of rows specified in the

AUTOSAVE option of the SET command has been entered and when you terminate the
INPUT command.

■ You can specify the COMMIT WORK option at anytime during INPUT command
processing to commit rows entered since the last COMMIT WORK was processed. For
example:

```
4> ('CharValue',NumericValue);
5> COMMIT WORK;
6>
```

■ You can specify an option called ROLLBACK WORK to back out rows entered since the
last COMMIT WORK was processed.

Using the SELECT Statement

When you enter a SELECT statement through ISQL, you can browse through the query result
at the terminal, print the query result, and/or edit the query result. The files used for these
activities are illustrated in Figure 3-1.
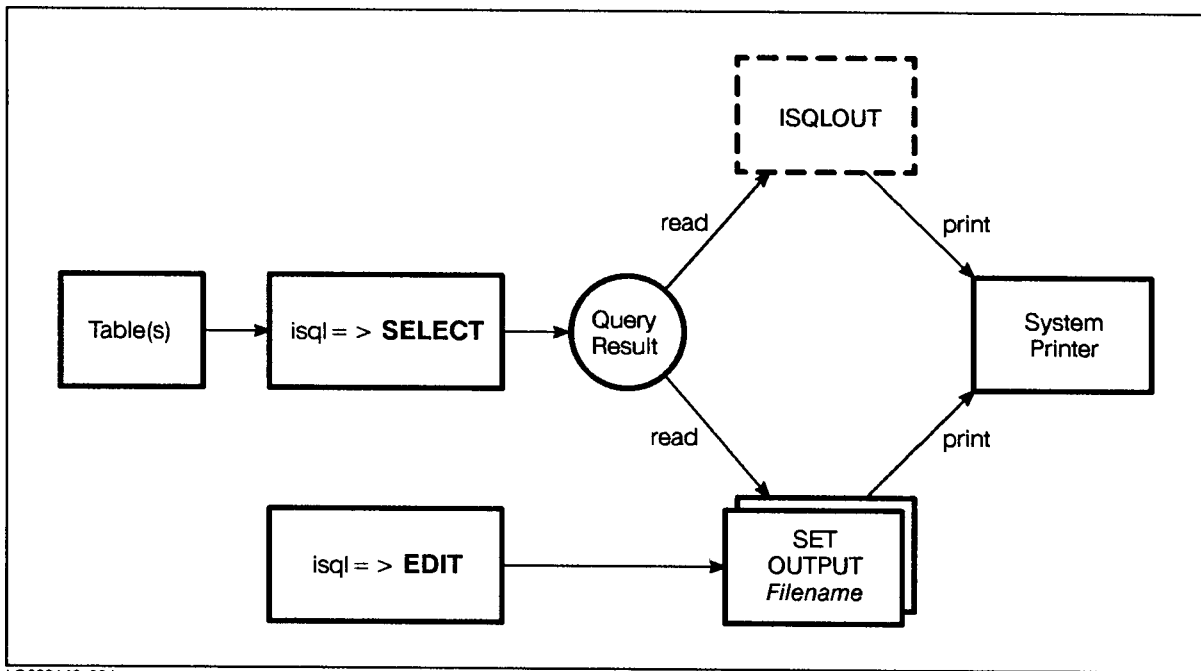
LG200142_004

Figure 3-1. Files Used for SELECT Command

ISQL puts the query result into the file named in the SET command with the OUTPUT
option. A file named *isqlout* is used if a SET OUTPUT command has not been issued to
name another file. The file isqlout is purged when the terminal display of the query result is
terminated. If you want to save a query result, change the name of the output file before you
enter the SELECT statement:

```
isql=> SET OUTPUT KeepFile;
isql=>
```

Ensure that you have write permission in the directory where isqlout or the newly named output file resides. The following describes how three additional SET options affect the query result:

■ The FRACTION option determines the number of decimal digits displayed in the query result for data of types FLOAT, REAL, and DECIMAL. The value of this option is initially 2.

■ The NULL option determines the character that ISQL displays when a null value occurs. The value of this option is initially a blank.

■ The PAGEWIDTH option determines the maximum length of each row ISQL puts into the output file. The initial setting of this option is its maximum value: 32767 bytes. Any rows in the query result that are longer than the PAGEWIDTH length are truncated.

Use the following commands to set options and issue a SELECT statement:

```
isql=> SET FRACTION 0; SET NULL ?;SET EDITOR /usr/bin/vi;
isql=> SELECT PARTNUMBER,UNITPRICE,DELIVERYDAYS
>FROM PURCHDB.SUPPLYPRICE;
```

Once the query result is in the output file, it is displayed on the terminal as follows:

```
SELECT PARTNUMBER,UNITPRICE,DELIVERYDAYS FROM PURCHDB.SUPPLYPRICE;
----------------+--------------+------------
PARTNUMBER      |UNITPRICE     |DELIVERYDAYS
----------------+--------------+------------
1123-P-01       |          450|          30
1123-P-01       |          475|          15
1123-P-01       |          550|          15
1123-P-01       |          475|          30
1123-P-01       |          500|          20
1123-P-01       |          525|          15
1133-P-01       |          180|          30
1133-P-01       |          200|          15
1133-P-01       |          220|          15
1133-P-01       |          195|          20
1143-P-01       |          180|          30
1143-P-01       |          185|          15
1143-P-01       |          175|          30
1143-P-01       |          180|          20
1153-P-01       |          210|          30
1153-P-01       |          220|          15
-----------------------------------------------------------------------------
First 16 rows have been selected.
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>,or e[nd]> b
```

ISQL displays 16 selected rows at a time. You can browse through the query result by responding to the prompt beneath the query result. The following are valid responses:

■ UP and DOWN scroll backward and forward, respectively, by half the number of displayed rows. You can also press the [Return] key for downward scrolling. The row count that ISQL displays beneath the query result is incremented each time you scroll DOWN.

- TOP and BOTTOM display the first and the last group of rows, respectively, of the query result. The row count that ISQL displays beneath the query result following a BOTTOM command reflects the total number of rows that qualify for the query.

- LEFT and RIGHT scroll the display left and right, respectively, by 40 screen columns at a time.

- PRINT *n* send *n* copies of the query result to the system printer; the default is one copy (n equals 1).

The b response, shown in the previous example, displays the last group of rows in the query result as shown below.

```
SELECT PARTNUMBER,UNITPRICE,DELIVERYDAYS FROM PURCHDB.SUPPLYPRICE;
----------------+--------------+------------
PARTNUMBER      |UNITPRICE     |DELIVERYDAYS
----------------+--------------+------------
1733-AD-01      |          255|          15
1823-PT-01      |          435|          20
1823-PT-01      |          450|          15
1823-PT-01      |          450|          15
1833-PT-01      |         1985|          20
1833-PT-01      |         1990|          15
1923-PA-01      |           70|          15
1923-PA-01      |           80|          20
1923-PA-01      |           70|          20
1923-PA-01      |           75|          10
1933-FD-01      |          565|          20
1933-FD-01      |          585|          15
1933-FD-01      |          600|          10
1933-FD-01      |          590|          15
1933-FD-01      |          585|          20
1943-FD-01      |          575|????????????

--------------------------------------------------------------------------
Number of rows selected is 69.
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd]> e

isql=>
```

The values in the UNITPRICE column are displayed without decimal digits because the FRACTION option of the SET command was 0 when the SELECT statement was executed. The last value in the DELIVERYDAYS column is displayed as several question marks, because the NULL option of the SET command was ? when the SELECT statement was executed and the value is null.

Because the OUTPUT option of the SET command was KeepFile when the SELECT statement was executed, the query result can be edited. You can use the ISQL EDIT command to access the query result. In the following example, the EDIT command invokes vi because the current EDITOR option of the SET command is /usr/bin/vi:

```
isql=> EDIT KeepFile;

SELECT PARTNUMBER,UNITPRICE,DELIVERYDAYS FROM PURCHDB.SUPPLYPRICE;
---------------+-------------+------------
PARTNUMBER     |UNITPRICE    |DELIVERYDAYS
---------------+-------------+------------
1123-P-01      |         450|          30
1123-P-01      |         525|          15
.
.
.
:q

isql=>
```

TID Function

Each row (tuple) in an ALLBASE/SQL table is stored at a database address on disk. This unique address is called the tuple identifier or TID. When using the TID function in a SELECT statement, specify it as you would a column name. For example:

```
isql=> SELECT TID(), PARTNUMBER FROM PURCHDB.SUPPLYPRICE;
```

The TID and partnumber for each row in the PurchDB.SupplyPrice table are displayed on the terminal as follows:

```
SELECT TID(), PARTNUMBER FROM PURCHDB.SUPPLYPRICE;
-----------+----------------
TID        |PARTNUMBER
-----------+----------------
    1:1:0|1123-P-01
    1:1:1|1123-P-01
    1:1:2|1123-P-01
    1:1:3|1123-P-01
    1:1:4|1123-P-01
    1:1:5|1123-P-01
    1:1:6|1133-P-01
    1:1:7|1133-P-01
    1:1:8|1133-P-01
    1:1:9|1133-P-01
   1:1:10|1143-P-01
   1:1:11|1143-P-01
   1:1:12|1143-P-01
   1:1:13|1143-P-01
   1:1:14|1153-P-01
   1:1:15|1153-P-01
-------------------------------------------------------------------
 First 16 rows have been selected.
 U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd] >
```

A TID consists of eight bytes of binary data and has the following format:

Table 3-1. SQLTID Data Internal Format

Content	Byte Range
Version Number	1 through 2
File Number	3 through 4
Page Number	5 through 7
Slot Number	8

ISQL parses and displays the TID as three fields separated by colons. The version number is not displayed. In the following example, the file number is 1, the page number is 2, and the slot number is 133.

```
1:2:133
```

When using the TID in a WHERE clause, the input parameter must be a constant. Only the equal and not equal comparison operators are supported. The version number field of the TID is optional, but if specified, must always be 0. The following example selects the row from PurchDB.Parts with a TID of 3:4:15.

```
isql=> SELECT * FROM PURCHDB.PARTS WHERE TID() = 3:4:15;
```

Using the DELETE Statement

The SQL DELETE statement lets you delete groups of rows from a table. Here is an example:

```
isql=> DELETE FROM PurchDB.Parts
> WHERE SalesPrice > 500;
```

All rows with a SalesPrice greater than $500 are deleted.

Using the UPDATE Statement

The SQL UPDATE statement lets you modify groups of rows in a table, as in the following example:

```
isql=> UPDATE PurchDB.Parts
> SET SalesPrice = 1.2*SalesPrice;
```

All prices are increased by 20 percent.

Using the INFO Command

The INFO command allows you to display the definition of each column in a table or view.

```
isql=> INFO PurchDB.Parts;


                                      Nulls            Case
Column Name    Data Type (length)     Allowed Language Sensitive
----------------------------------------------------------------
PARTNUMBER     Char      (  16)         NO    n-computer   YES
PARTNAME       Char      (  30)         YES   n-computer   YES
SALESPRICE     Decimal   (  10,   2)    YES

isql=>
```

Unloading and Loading Tables

ISQL provides the following commands for loading and unloading data:

- UNLOAD copies data from tables into a file.
- LOAD provides higher-volume data loading than the INPUT or INSERT. commands by providing bulk inserts using data stored in a file.

Like the other ISQL commands, these commands cannot be used from within an application program.

Using the UNLOAD Command

You can unload all of a table to an output file by specifying a table name, or you can unload selected data by specifying a SELECT statement with the UNLOAD command. Also, the data can be unloaded in sorted order if you use the GROUP BY or ORDER BY clause as part of the SELECT statement. For complete information on the SELECT statement, refer to the *ALLBASE/SQL Reference Manual*. You specify the format of the output file as either external or internal.

External Format

You unload a table to an external data file using the external format for the following reasons:

- To use the data file in an application other than ALLBASE/SQL.
- To change column names, sizes, or data types when reloading.
- To edit the file.
- To do maintenance tasks, such as splitting columns.

When you use the EXTERNAL option of the UNLOAD command, ISQL creates an external file and a description file. The external file contains the data; the description file describes the location, length, and format of the data in the external file. The external and description files can be read by an editor, an application program, or ISQL (when you use the LOAD command with the EXTERNAL option). Application programs and ISQL use the description file to interpret the format of the data in the external data file.

Note	You can use the UNLOAD command with the EXTERNAL option to unload LONG column values. For LONG columns, the data is placed in the output file or device that was specified at the time the data was inserted into the table (or at the time the column was last updated). The output file or device name is placed in the external file.

You can unload data from a table in the following ways:

- With the prompting mode of the UNLOAD command.
- Directly from the isql prompt.
- By using a command file.

When you are using prompting mode as shown below, ISQL displays the attributes of each column you are unloading and prompts you to describe how values from this column should appear in the external file.

```
isql=> UNLOAD;

Output file format (i[nternal] or e[xternal])> EXTERNAL;
Output file name> EParts;
TableName or "SelectStatement"> PurchDB.Parts;
Description file name> DParts;
Column PARTNUMBER, Char (16):
Output length> 16;
Column PARTNAME, Char (30):
Output length> 30;
Null indicator> ?;
Column SALESPRICE, Decimal (10,2):
Output length> 11;
Fraction length> 2;
Null indicator> ?;
Command in progress.
Number of rows processed is 21

isql=>
```

In the case of LONG columns, you should enter the length of the output device name in response to the prompt for output length.

To unload the same file as in the example above, you can also enter the necessary information directly on the command line:

```
isql=> UNLOAD TO EXTERNAL EParts FROM PurchDB.Parts
> DParts 16 30 ? 11 2 ?;
```

Note that the keywords TO and FROM are necessary in addition to the responses given if you had been prompted.

Similarly, you could create a command file containing the following lines:

```
unload to external
EParts from PurchDB.Parts
DParts 16 30 ? 11 2 ?
```

To execute the command file, use the START command. For example, if the file is named MyFile, issue the following command:

```
isql=> START MyFile;
```

The description file is automatically created and contains a record for each column unloaded. The record describes where and in what format the data exists in the external file. You can use your editor to view this file. The description file has seven columns containing the following information about the external data file:

```
Characters  1 - 20:  Column name
Character        25:  Data type code
Characters 30 - 40:  Output length
Characters 41 - 50:  Fraction Length
Characters 51 - 60:  Starting location of the data
Characters 61 - 70:  Starting location of length of VARCHAR data
Character        75:  Character chosen for the NULL indicator
```

You may unload the TID (tuple identifier) data by specifying the SELECT statement:

```
isql=> UNLOAD;

Output file format (i[nternal] or e[xternal])> EXTERNAL;
Output file name> TidOut;
TableName or "SelectCommand"> "SELECT Tid() FROM PurchDB.Parts";
Description file name> TidDesc;
Column TID, Tuple ID:
Output length> 16;
Command in progress.
Number of rows processed is 22

isql=>
```

Note	Although the UNLOAD command allows you to copy TID data into a file, you may not use the LOAD command to load the TID data back into a table. The TID is a unique identifier generated internally by ALLBASE/SQL, and may not be assigned by user programs.

Internal Format

The internal format specified in the following example is understandable only by ALLBASE/SQL. It is faster than the EXTERNAL option, because the data is not converted from internal to ASCII format. Files in internal format are read by ISQL when you use the LOAD command with the INTERNAL option.

```
isql=> UNLOAD;

Output file format (i[nternal] or e[xternal])> INTERNAL;
Output file name> IntFile;
TableName or "SelectCommand"> PurchDB.Vendors;
Command in progress.
Number of rows processed is 15

isql=>
```

You cannot use the INTERNAL option to unload LONG column data.

Using the LOAD Command

You load data stored in a file into a table by using the LOAD command. The file is a data file
in external or internal format.

External Format

The EXTERNAL option of the LOAD command is used to load external data files into a
database table. The external data file can be one of the following:

- The external file produced when the EXTERNAL option of the UNLOAD command is
  used.
- A file created with an editor.
- The output of an application program.
- A file created on an IBM mainframe.

You can load data into a table in the following ways:

- Using the external data and description files produced with the UNLOAD EXTERNAL
  command.
- Creating an external data file and using the prompting mode of the LOAD command.
- Creating an external data file and entering the LOAD EXTERNAL command from the isql
  prompt.
- Creating an external data file and using a command file.

To load the data, which was unloaded with the EXTERNAL option in the previous example,
you specify the external data file name, the table where the data is to be loaded, and the
description file name in the LOAD FROM EXTERNAL command as follows:

```
isql=> LOAD FROM EXTERNAL EParts to Purchdb2.Parts USING DParts;
```

If you want to load data that was not produced with the EXTERNAL option of the UNLOAD
command, you need to create the external data file and provide a description of the data to be
loaded as part of the LOAD command.

You can create the external data file with the editor as shown below:

```
isql=> EDIT;
```

The editor named in the EDITOR option of the SET command is invoked.

```
1145            Abacus                   15.50
1167            Vacuum tube              7.95
260097          Analytic engine          9000.50
⋮
:wq EParts2

isql=>
```

String data does not need to be enclosed in quotation marks. You do not need to align numeric values within a column, and you can extend beyond column 72.

To provide a description of the data file, you can use the LOAD command in prompting mode to provide ISQL with the location and length of the data, and the character used for the null indicator. To obtain this information you can use the INFO command. For example:

```
isql=> LOAD;

File format (i[nternal] or e[xternal])> EXTERNAL;
Input file name> EParts2;
Table name> OldParts;
Enter information as requested for each column to be loaded.
Enter END for the column name when finished.
Column name> PartNumber;
Starting location> 1;
Column length> 16;
Column name> PartName;
Starting location> 17;
Column length> 30;
Null representation> ?;
Column name> SalesPrice;
Starting location> 47;
Column length> 10;
Null representation> ?;
Column name> END;
Load depending on value in input record (Y/N)> Y;
Starting location of select field> 1;
Select field pattern> 1;
Command in progress
Number of rows read is 21
Number of rows processed is 21
COMMIT WORK to save to DBEnvironment

isql=>
```

In the example above, you can only load data from records that contain a specific character string (Select field pattern) beginning at a specific location (Starting location of select field) in the input records. In this example, only parts having numbers that start with 1 are loaded.

You can also specify only a range of records using the PARTIAL option. Refer to the LOAD command in chapter 4, "ISQL Commands," for a description of the PARTIAL option.

To load the above data into a table, you enter the LOAD FROM EXTERNAL command and include the following information:

■ External data file name

■ Name of the target file

■ Responses which you would give if you were in prompting mode

For example:

```
isql=> LOAD FROM EXTERNAL EParts to PurchDB.Parts
> PARTNUMBER     1     16
> PARTNAME      17     30    ?
> SALESPRICE    47     10    ?
> END
> Y 1 1;
```

The indicated column name, starting location, length, and the character used as the null character are specified. The command is terminated by the word END, a semicolon, and a *Y* or *N*. The N response to the question indicates that you want to load *all* the rows in the external data file. A Y response requires that you include the pattern location (the column number where the pattern begins) and the pattern.

Similarly, if you want to use a command file, it would look like this:

```
LOAD FROM EXTERNAL EParts to PurchDB.Parts
PARTNUMBER     1     16
PARTNAME      17     30    ?
SALESPRICE    47     10    ?
END
Y 1 1;
```

To execute the command file use the START command as follows:

```
isql=> START MyFile;
```

Internal Format

If you used the INTERNAL option of the the UNLOAD command, you specify it again when loading the data. For example:

```
isql=> LOAD;

File format (i[nternal] or e[xternal])> INTERNAL;
Input file name> IntFile;
Table name> PurchDB.Vendors;
Number of rows read is 15
Number of rows processed is 15
COMMIT WORK to save to DBEnvironment.
isql=>
```

Setting the AUTOCOMMIT Option

Modified or newly loaded data is written to the log buffer. When a COMMIT WORK is processed, the data is moved from the log buffer to the log file and the changes to the DBEnvironment are made permanent. When nonarchive mode (the default) is set, the nonarchive log file space is reclaimed.

As in the case of the INPUT command, ISQL processes the COMMIT WORK statement based on the AUTOCOMMIT option value of the SET command. If AUTOCOMMIT is ON, ISQL will continue to accept rows for input until the load buffer is filled, even if the number of rows specified in the AUTOSAVE option has been exceeded. Once the load buffer is full, a check is made to see if the number of rows specified in the AUTOSAVE option has been reached or exceeded. If this is the case ISQL processes a COMMIT WORK statement.

If AUTOCOMMIT is OFF (the default setting), the rows are committed only when you enter the COMMIT WORK statement. When using nonarchive mode, use the AUTOCOMMIT command with the option value set to ON to ensure that the nonarchive log file (a circular file) is not overwritten before you process the COMMIT WORK statement.

Using Modules

A **module** is a set of one or more preprocessed statements that are stored in the DBEnvironment. Each stored statement is known as a **section**. ALLBASE/SQL application programs usually access modules that are stored by the preprocessor. You can also create modules interactively with the PREPARE statement in ISQL.

Installing a Module

An **installable module**, which is created by the ALLBASE/SQL preprocessor before source code is compiled, contains information that ALLBASE/SQL uses to execute the application program's SQL statements at run time. The program can only access the DBEnvironment in which its corresponding module is stored.

Sometimes you need to transfer a user-written application from one DBEnvironment to another. For instance, an application created on a development machine may need to be transferred to a production system. You would need to unload the data from one DBEnvironment and then load it into the new one.

Additionally, to use an application program in a DBEnvironment different from that in which it was developed, you must move the executable application program file and the file containing the installable module to the machine containing the target DBEnvironment.

To store the installable module in the new DBEnvironment, use the ISQL INSTALL command:

```
isql=> INSTALL;

File name> someprog.sqlm;
Name of module in this file:  Pgmr1.someprog
Number of sections installed:  6
COMMIT WORK to save to DBEnvironment.

isql=>
```

You can use the ISQL LIST INSTALL command to identify the names of the modules
contained in an installable module file. For example:

```
isql=> LIST INSTALL;

File name> someprog.sqlm;
Name of module in this file:  Pgmr1.someproga
Name of module in this file:  Pgmr1.someprogb

isql=>
```

To copy a module in a DBEnvironment to an installable module file, use the ISQL EXTRACT
command:

```
isql=> EXTRACT MODULE CEX9 INTO cex9.sqlm;

Extract    6 sections from module CEX9.

Number of modules extracted: 1.

isql=>
```

Note	A module stored in a DBEnvironment by a preprocessor, or a module installed using the ISQL INSTALL command, should *never* be executed directly from ISQL by typing the name of the module. The module should be executed by running the executable program file.

For more information on using ALLBASE/SQL programmatically, refer to the
ALLBASE/SQL application programming guide for the language you are using or to the
appropriate ALLBASE/SQL application programming bulletin.

Modules can be created by the preprocessor or the PREPARE statement. The preprocessor
inserts a single module into the module installation files. Several module installation files may
be joined into a single installation file with the cat command. The PREPARE statement
has an interactive option which allows creation of a module through ISQL. The EXECUTE
statement can then be used to cause ALLBASE/SQL to execute the statement that was
stored with the PREPARE statement. A module created by using the PREPARE statement
with ISQL cannot be executed by an application with the preprocessor. For more information
on the PREPARE and EXECUTE statements, refer to the *ALLBASE/SQL Reference
Manual*.

4

ISQL Commands

This chapter describes all the ISQL commands, giving syntax and examples for each and authorization requirements for commands that result in database access. The commands are presented in alphabetical order; those related to query results are defined under "SELECTSTATEMENT."

The size of commands submitted at a terminal or from an ISQL command file is unlimited. Commands submitted through the ISQL command buffer cannot exceed 32K bytes; the LOAD and UNLOAD size is unlimited.

Table 4-1 groups the commands you can execute from ISQL into functional categories. Refer to the *ALLBASE/SQL Reference Manual* for a description of those in the category named SQL statement. Refer to the *ALLBASE/SQL Database Administration Guide* for SQLGEN and SQLUtil commands.

Table 4-1. ISQL Command Summary

Group	Category	Command	Command Use
DBEnvironment access	SQL statements	Refer to the *ALLBASE/SQL Reference Manual.*	Define, access, and maintain databases.
	Column information	`INFO`	Display column definitions.
	Data utility	`INPUT`	Insert rows into a table.
		`LOAD`	Insert rows into a table from a file.
		`UNLOAD`	Copy rows from one or more tables to a file.
	Application program support	`EXTRACT`	Copy a stored module into an installable module file.
		`INSTALL`	Store a preprocessor-generated module in a DBEnvironment.
		`LIST INSTALL`	Identify the module in a file created by an ALLBASE/SQL preprocessor.

Table 4-1. ISQL Command Summary (continued)

Group	Category	Command	Command Use
Command support	Query results	UP	Display previous rows in a query result.
		DOWN	Display next rows in a query result.
		TOP	Display the group of selected rows of a query result.
		BOTTOM	Display the last group of selected rows of a query result.
		LEFT	Display previous screen columns in a query result.
		RIGHT	Display next screen columns in a query result.
		PRINT	Print a query result.
		PAUSE	Suspend command file processing to enable review of a query result.
		END	Terminate query result display.
	File management	EDIT	Invoke an editor for creating or changing a file.
		ERASE	Purge a file.
		LIST FILE	Display a file.
		RECALL FILE	Display a file and store its contents in command buffer.
		RENAME	Change the name of a file.
		STORE	Save the contents of the command buffer in a file.
	Command files	START	Execute a command file.
	Command buffer	CHANGE	Modify the contents of the command buffer.
		HOLD	Put command(s) into the command buffer.

Table 4-1. ISQL Command Summary (continued)

Group	Category	Command	Command Use
Command support (continued)	Command buffer (continued)	`RECALL CURRENT`	Display current contents of the command buffer.
		`RECALL FILE`	Displays a file and stores its contents in the command buffer.
		`RECALL HISTORY`	Place a command from the command history buffer into the command buffer and display it.
		`START`	Execute command(s) in the command buffer or a file.
		`STORE`	Save the contents of the command buffer in a file.
	Command history buffer	`DO`	Execute a command in the command history buffer.
		`LIST HISTORY`	Display commands in the command history buffer.
		`RECALL HISTORY`	Place a command from the command history buffer into the command buffer and display it.
		`REDO`	Edit a command in the command history buffer.
	ISQL environment	`LIST SET`	Display SET option values.
		`SET`	Set operational options, such as display line length, to specific values.
	Command information	`HELP`	Display description, syntax, and example of SQL or ISQL commands.
ISQL termination		`END` *or* `EXIT`	Terminate an ISQL session.
System access		`SYSTEM`	Execute HP-UX command(s).
SQLGEN invocation		`SQLGEN`	Execute SQLGEN commands.
SQLUtil invocation		`SQLUTIL`	Execute SQLUtil commands.

CHANGE

The CHANGE command modifies the first occurrence or all occurrences of a string in the command buffer.

Scope

ISQL only.

ISQL Syntax

C[HANGE] *Delimiter OldString Delimiter NewString Delimiter* [@]

Parameters

Delimiter	is a one-byte nonblank character delimiting *OldString* and *NewString*.
OldString	is the string in the command buffer to be replaced. Its maximum length is 32K bytes.
NewString	is the string to replace the *OldString*. Its length does not have to equal that of *OldString*. Its length can range from 0 to 32K bytes.
@	is specified to change all occurrences of *OldString*. If omitted, only the first occurrence of *OldString* is changed. If @ is the line-kill character on your system, you must precede the @ with the current HP-UX escape character.

Description

■ If *Delimiter*, *OldString*, or *NewString* contains one of the following special characters, precede the special character with the escape character defined with the ESCAPE option of the SET command:

    '
    "
    &
    ;
    SET ESCAPE *Character* (initial setting = \)

■ In prompting mode, do not enter delimiters. For example:

    isql=> CHANGE;

    Old string> *OldString*;
    New string> *NewString*;
    Change all occurrences (Y/N)> Y

Changed command is displayed.

Example

```
isql=> RECALL CURRENT;

SELECT Name,Owner FROM System.Table WHERE Type = 'O';

isql=> CHANGE |\'||@;
```

When nothing is inserted for the new string, the old string is deleted.

```
SELECT Name,Owner FROM System.Table WHERE Type = O;

isql=> CHANGE |Owner|Nrows,Avglen|;

SELECT Name,Nrows,Avglen FROM System.Table WHERE Type = O;

isql=> CHANGE |,Avglen||;

SELECT Name,Nrows FROM System.Table WHERE Type = O;

isql=> START;
```

DO

The DO command lets you execute any of the commands in the command history buffer.

Scope

ISQL only.

ISQL Syntax

$$\text{DO} \begin{bmatrix} CommandNumber \\ CommandString \end{bmatrix}$$

Parameters

CommandNumber   identifies one of the commands in the command history buffer. The command history buffer holds the fifty most recently submitted commands, numbered 1 through 50. The *CommandNumber* of the most recently submitted command is 1.

CommandString   identifies the most recent command in the command history buffer that begins with the *CommandString*.

Description

■ You can use the LIST HISTORY @ command to identify the *CommandNumber* associated with each command currently in the command history buffer.

■ If no parameter is specified, the most recently submitted command is assumed.

■ DO cannot be used from a command file or the command buffer.

Example

```
isql=> LIST HISTORY @;

1     SELECT SalesPrice FROM PurchDB.Parts;
2     INFO PurchDB.Parts;
3     EDIT MyFile;
 .
 .
 .

isql=> DO 2;
```

The "INFO PurchDB.Parts" command is executed because it is the second command in the command history buffer.

```
isql=> DO;
```

The "INFO PurchDB.Parts" command is executed again because it is the most recent command in the command history buffer.

```
isql=> DO ED;
```

The "EDIT MyFile" command is executed because it is the most recent command in the command history buffer that begins with the string "ED".

EDIT

The EDIT command invokes the editor named using the SET EDITOR option for creating or updating a file.

Scope

ISQL only.

ISQL Syntax

$$\text{ED}\,[\,\text{IT}\,]\,[\,FileName\,]$$

Parameters

FileName   identifies the file you want to edit. Name qualification follows HP-UX conventions:

$[PathName/]FileName$

Unless you specify an absolute path name, ISQL assumes that any path name you specify is relative to your current working directory.

Example

```
isql=> LIST SET EDITOR;
ed[itor]              - /usr/bin/vi

isql=> EDIT;
~
~
~
UPDATE STATISTICS FOR TABLE PurchDB.SupplyPrice;
UPDATE STATISTICS FOR TABLE PurchDB.Orders;
:wq Daily
"Daily" [New file] 2 lines, 94 characters
isql=> SET EDITOR /usr/bin/ex;
isql=> EDIT;
:e Daily
"Daily" 2 lines, 94 characters
:1,$#
1  UPDATE STATISTICS FOR TABLE PurchDB.SupplyPrice;
2  UPDATE STATISTICS FOR TABLE PurchDB.Orders;
:1,s/SupplyPrice/OrderItems/
UPDATE STATISTICS FOR TABLE PurchDB.OrderItems:
:wq
"Daily" 2 lines, 94 characters
isql=>
```

END

The END command terminates an ISQL session and returns you to HP-UX.

Scope

ISQL only.

ISQL Syntax

EN$\left[\text{D}\right]$

Description

■ If you have a DBEnvironment transaction in progress when you issue the END command, ISQL prompts you as follows:

    A transaction is in progress. Commit work (Y/N)>

Enter N to roll back any DBEnvironment change(s) you made since signing on or since the latest COMMIT WORK. ISQL automatically processes a ROLLBACK WORK statement, and your changes are undone.

Enter Y to commit any DBEnvironment change(s) you made since signing on or since the latest COMMIT WORK. ISQL automatically processes a COMMIT WORK statement, and any of your changes become permanent.

■ After any transaction has ended, the DBE session is automatically terminated. ISQL then issues a RELEASE or STOP DBE statement on your behalf.

■ END has the same effect as EXIT.

Example

    isql=> END;

    $

ERASE

The ERASE command deletes a file.

Scope

ISQL only.

ISQL Syntax

$$\text{ER}\big[\,\text{ASE}\,\big]\,FileName$$

Parameters

FileName   identifies the file to be erased. Name qualification follows HP-UX conventions:

$$[\,PathName/\,]\,FileName$$

Unless you specify an absolute path name, ISQL assumes that any path name you specify is relative to your current working directory.

Description

■ In prompting mode, ISQL prompts you for a file name, as shown here:

```
isql=> ERASE;
File name> FileName;
isql=>
```

Example

```
isql=> ERASE Old;
isql=>
```

EXIT

The EXIT command terminates an ISQL session and returns you to HP-UX.

Scope

ISQL only.

ISQL Syntax

EX[IT]

Description

■ If you have a DBEnvironment transaction in progress when you issue the EXIT command, ISQL prompts you as follows:

```
A transaction is in progress.  Commit work (Y/N)>
```

Enter N to roll back any DBEnvironment change(s) you made since signing on or since the latest COMMIT WORK. ISQL automatically processes a ROLLBACK WORK statement, and your changes are undone.

Enter Y to commit any DBEnvironment change(s) you made since signing on or since the latest COMMIT WORK. ISQL automatically processes a COMMIT WORK statement, and your change(s) become permanent.

■ After any transaction has been ended, the DBE session is automatically terminated. ISQL then issues a RELEASE or STOP DBE statement on your behalf.

■ EXIT has the same effect as END.

Example

```
isql=> EXIT;

$
```

EXTRACT

The EXTRACT command copies modules from a DBEnvironment into an installable module file.

Scope

ISQL only.

ISQL Syntax

$$\text{EXTRACT} \left\{ \begin{array}{l} \text{MODULE } [\,Owner.\,]ModuleName \,[\,,\,\ldots\,] \\ \text{SECTION } [\,Owner.\,]ModuleName(SectionNumber) \,[\,,\,\ldots\,] \\ \text{ALL MODULES} \end{array} \right\}$$

$$[\text{NO SETOPTINFO}]\text{INTO } FileName$$

Parameters

[*Owner.*]*ModuleName*   identifies the module containing sections to be extracted.

SectionNumber   specifies the section contained in the module.

FileName   identifies the file containing the installable modules. If the file already exists, it is overwritten.

Description

■ The EXTRACT command does not remove a module from the DBEnvironment. It copies the module into an installable module file.

■ The NO SETOPTINFO clause prevents access plans specified by the SETOPT statement from being used by the sections placed in the installable module file.

■ Modules are installed with the INSTALL command and deleted with the DROP MODULE statement. Use the LIST INSTALL command to view the names of modules in an installable module file.

■ Query the SYSTEM.SECTION view to identify the module name and section numbers.

Authorization

You can execute this statement if you have DBA or OWNER authority for the module.

Example

```
isql=> EXTRACT MODULE CEX9 INTO cex9.sqlm;

Extract     6 sections from module CEX9.

Number of modules extracted: 1.

isql=>
```

HELP

The HELP command displays information about ISQL commands and SQL statements.

Scope

ISQL only.

ISQL Syntax

$$\text{HE}\big[\text{LP}\big] \left\{ \begin{array}{l} \text{@} \\ \textit{SQLstatement} \\ \textit{ISQLcommand} \end{array} \right\} \left[\begin{array}{l} \text{D}\big[\text{ESCRIPTION}\big] \\ \text{S}\big[\text{YNTAX}\big] \\ \text{E}\big[\text{AMPLE}\big] \end{array} \right]$$

Parameters

@ is specified to obtain a list of the SQL statements and a list of the ISQL commands. If @ is the line-kill character on your system, you must precede the @ with the current HP-UX escape character.

SQLstatement is any unabbreviated SQL statement. This consists of the SQL verb and the command keywords that follow.

ISQLcommand is any unabbreviated ISQL command. This consists of the ISQL verb and the command keywords that follow.

DESCRIPTION specifies that the information displayed is limited to the description of the requested *SQLstatement* or *ISQLcommand*.

SYNTAX specifies that the information displayed is limited to the syntax of the requested *SQLstatement* or *ISQLcommand*.

EXAMPLE specifies that the information displayed is limited to examples of the requested *SQLstatement* or *ISQLcommand*.

Description

- If you specify a command verb that is used in more than one command, ISQL lists all the commands that qualify and prompts you to enter the entire command.

- In prompting mode, ISQL prompts you for a command name.

Example

```
isql=> HELP @;
```

In the following list of available HELP statements, those marked by * can NOT be executed from ISQL.

HELP

```
===== SQL Statements =====

  ADD DBEFILE                   EXECUTE IMMEDIATE
  ADD TO GROUP                  EXECUTE PROCEDURE
 *ADVANCE                      *FETCH
  ALTER DBEFILE                *GOTO
  ALTER TABLE                   GRANT
 *BEGIN                        *IF
  BEGIN ARCHIVE                *INCLUDE
 *BEGIN DECLARE SECTION         INSERT
  BEGIN WORK                    LOCK TABLE
  CHECKPOINT                    LOG COMMENT
 *CLOSE                        *OPEN
  COMMIT ARCHIVE                PREPARE
  COMMIT WORK                  *PRINT
  CONNECT                       RAISE ERROR
  CREATE DBEFILE               *REFETCH
  CREATE DBEFILESET             RELEASE
  CREATE GROUP                  REMOVE DBEFILE
  CREATE INDEX                  REMOVE FROM GROUP
  CREATE PARTITION              RESET
  CREATE PROCEDURE             *RETURN
  CREATE RULE                   REVOKE
  CREATE SCHEMA                 ROLLBACK WORK
  CREATE TABLE                  SAVEPOINT
  CREATE TEMPSPACE              SELECT
  CREATE VIEW                   SET CONNECTION
 *DECLARE                       SET CONSTRAINTS
 *DECLARE CURSOR                SET DEFAULT DBEFILESET
  DELETE                        SET DML ATOMICITY
 *DELETE WHERE CURRENT          SET MULTITRANSACTION
 *DESCRIBE                      SET PRINTRULES
  DISABLE AUDIT LOGGING         SET SESSION
  DISABLE RULES                 SET TRANSACTION
  DISCONNECT                    SET USER TIMEOUT
  DROP DBEFILE                  SETOPT
  DROP DBEFILESET              *SQLEXPLAIN
  DROP GROUP                    START DBE
  DROP INDEX                    START DBE NEW
  DROP MODULE                   START DBE NEWLOG
  DROP PARTITION                STOP DBE
  DROP PROCEDURE                TERMINATE USER
  DROP RULE                     TRANSFER OWNERSHIP
  DROP TABLE                    TRUNCATE TABLE
  DROP TEMPSPACE                UPDATE
  DROP VIEW                     UPDATE STATISTICS
  ENABLE AUDIT LOGGING         *UPDATE WHERE CURRENT
  ENABLE RULES                  VALIDATE
 *END DECLARE SECTION          *WHENEVER
  EXECUTE                      *WHILE
```

```
===== ISQL Commands =====

   CHANGE          GENPLAN        LIST HISTORY    SET
   DO              HELP           LIST INSTALL    SQLGEN
   EDIT            HOLD           LIST SET        SQLUTIL
   END             INFO           LOAD            START
   ERASE           INPUT          RECALL          STORE
   EXIT            INSTALL        REDO            SYSTEM
   EXTRACT         LIST FILE      RENAME          UNLOAD
```

isql=> help help;

SCOPE: ISQL Only

The HELP command displays information about ISQL or SQL commands.

```
==========
  SYNTAX
==========


      {@              } [D[ESCRIPTION]]
HE[LP]{SQLStatement} [S[YNTAX    ]]
      {ISQLCommand } [E[XAMPLE    ]]
```

The conventions used in the syntax diagrams displayed by the
HELP command are as follows:

UPPERCASE   Commands and keywords are shown in uppercase characters.
        The characters must be entered in the order shown.
        However, you can enter the characters in either uppercase
        or lowercase.

Lowercase   A capitalized, lowercase word represents a parameter or
        argument that you must replace with the actual value.

punctuation   Punctuation characters (other than brackets, braces,
        vertical bars, and ellipses) must be entered exactly
        as shown.

{ }   Braces enclose required elements. When several elements
        are stacked within braces, you must select one. In the
        following example, you must select either ON or OFF.

          COMMAND {ON }
            {OFF}

[ ]   Brackets enclose optional elements. When several elements
        are stacked within brackets, you can select one or none of
        the elements. In the following example, you can select
        OPTION or parameter or neither.

```
      COMMAND [OPTION   ]
         [parameter]
```

[...]   Horizontal ellipses enclosed in brackets indicate that
        you can repeatedly select the element(s) that appear
        within the immediately preceding brackets or braces.
        In the following example, you can select parameter zero
        or more times.

   [parameter] [...]

|...| Horizontal ellipses enclosed in vertical bars indicate
        that you can select more than one element within the
        immediately preceding brackets or braces. However,
        each particular element can only be selected once.
        In the following example, you must select A, AB, BA,
        or B. The elements cannot be repeated.

   {A} |...|
   {B}

```
===========
   EXAMPLE
===========
```

HELP @ ;

HELP input ;

isql=> help create;

Please specify help for one of the following commands:

```
  CREATE DBEFILE
  CREATE DBEFILESET
  CREATE GROUP
  CREATE INDEX
  CREATE PROCEDURE
  CREATE RULE
  CREATE SCHEMA
  CREATE TABLE
  CREATE TEMPSPACE
  CREATE VIEW
```

```
isql=> help do;
```

SCOPE: ISQL Only

The DO command lets you execute any of the commands in the command history
buffer.

```
==========
   SYNTAX
==========
```

```
DO [CommandNumber]
   [CommandString]
```

```
==========
   EXAMPLE
==========
```

```
DO ;
```

```
DO 2 ;
```

```
isql=> help add dbefile example;
```

```
==========
   EXAMPLE
==========
```

```
ADD DBEFILE ThisDBEFile TO DBEFILESET Miscellaneous ;
```

```
isql=> help add dbefile description;
```

SCOPE: ISQL or Application Programs

The ADD DBEFILE statement associates a DBEFile with a DBEFileSet.

```
isql=> help add dbefile syntax;
```

```
==========
   SYNTAX

==========
```

```
ADD DBEFILE DBEFileName TO DBEFILESET DBEFileSetName
```

HOLD

The HOLD command puts SQL and ISQL commands into the command buffer, overwriting the current contents of the command buffer.

Scope

ISQL only.

ISQL Syntax

$$\text{HO}\left[\text{LD}\right]\left\{ \begin{array}{l} SQLStatement \\ ISQLCommand \end{array} \right\}\left[EscapeCharacter; \left\{ \begin{array}{l} SQLStatement \\ ISQLCommand \end{array} \right\} \right]\left[\ \ldots \ \right]$$

Parameters

SQLStatement	is any SQL statement, which may contain one or more parameters.
ISQLCommand	is any ISQL command, which may contain one or more parameters.
EscapeCharacter	is the character currently defined with the SET command using the ESCAPE option. Use the escape character and a semicolon between commands when you put multiple commands into the command buffer.

Description

■ Identify parameters with an ampersand (&) and a number from 1 through 100. For example:

```
isql=>  HOLD SELECT &1 FROM &2 WHERE &3;
```

You substitute values for the parameters when you issue the START command. Parameters assigned the same number are all assigned the same value. Refer to the START command for more information.

■ When commands are in the command buffer, you can operate on them with the following ISQL commands:

CHANGE
RECALL CURRENT
START
STORE

■ The command buffer can hold as many as 32K bytes.

■ The following ISQL operations replace the contents of the command buffer:

HOLD
RECALL FILE
RECALL HISTORY

■ When scanning a HOLD command, ISQL deletes any escape character and treats the character following an escape character as a simple character. Therefore, if an escape character is to appear in the command buffer, you must use three escape characters instead of one (that is, the escape character and the following character must each be escaped). The escape character and semicolon are still required between commands when multiple commands are placed in the command buffer.

The following shows the correspondence between what you want to place in the command buffer and what you must type in with the HOLD command:

To put this in command buffer:	*Type this:*
SYSTEM cat isqlpro\; date;	HOLD system cat isqlpro\\\; date;
SYSTEM cat isqlpro; info t1;	HOLD system cat isqlpro\; info t1;
SELECT * from t1 where c = '\\';	HOLD select * from t1 where c = '\\\\';

Example

```
isql=> RECALL CURRENT;

Command buffer currently empty.  (DBERR 1)

isql=> HOLD SELECT &1 FROM &2 WHERE &3;
isql=> RECALL CURRENT;

SELECT &1 FROM &2 WHERE &3;

isql=>
```

INFO

The INFO command displays definitions of the columns of a table or view.

Scope

ISQL only.

ISQL Syntax

$$\text{IN}\big[\text{FO}\big]\left\{\begin{array}{l} \big[\,Owner.\,\big]\,TableName \\ \big[\,Owner.\,\big]\,ViewName \end{array}\right\}$$

Parameters

[*Owner.*]*TableName*   identifies a table in the DBEnvironment you are using.

[*Owner.*]*ViewName*   identifies a view in the DBEnvironment you are using.

Description

■ ISQL responds with the name of the language of the table and a display of its column definition.

■ The following information is displayed for each column in the table or view:

  □ Column name.
  □ Data type.
  □ In parentheses, length (in bytes) of CHAR or VARCHAR data, or precision and scale of DECIMAL, REAL, or FLOAT data.
  □ Indication (YES or NO) of whether null values are allowed.
  □ Name of the language for each CHAR or VARCHAR column.

■ You must establish a DBE session before you can use INFO. Use the SQL CONNECT or START DBE statement to establish a DBE session.

■ In prompting mode, ISQL prompts you for a table name as follows:

```
isql=> INFO;

Table name> TableName;  or  ViewName;
Table language = LanguageName


                                 Nulls              Case
Column Name       Data Type (length)   Allowed   Language   Sensitive
-----------------------------------------------------------------------

ColumnName        Type (length)        YES or NO  LangName   YES or NO

⋮
isql=>
```

Authorization

You must have authorization to select from the named table or view. Refer to the discussion of the SELECT statement in the *ALLBASE/SQL Reference Manual.*

Example

```
isql=> INFO System.Table;


                                             Nulls                   Case
Column Name        Data Type  (length)    Allowed  Language       Sensitive
-----------------------------------------------------------------------------
    NAME           Char       (   20)       NO     n-computer        YES
    OWNER          Char       (   20)       NO     n-computer        YES
    DBEFISET       Char       (   20)       NO     n-computer        YES
    TYPE           SmallInt                 NO
    RTYPE          SmallInt                 NO
    NUMC           Integer                  NO
    NUMI           Integer                  NO
    NUMIC          Integer                  NO
    NUMR           Integer                  NO
    NPAGES         Integer                  NO
    NROWS          Integer                  NO
    AVGLEN         Integer                  NO
    CTIME          Char       (   16)       NO     n-computer        YES
    USTIME         Char       (   16)       NO     n-computer        YES
    LANGUAGEID     SmallInt                 NO
    PARTITION      Char       (   20)       NO     n-computer        YES


isql=>
```

INPUT

The INPUT command inserts rows into an existing table.

Scope

ISQL only.

ISQL Syntax

$$\text{INP}\left[\text{UT}\right]\left\{\begin{array}{l}\left[\,Owner.\,\right]TableName\\\left[\,Owner.\,\right]ViewName\end{array}\right\}(\,ColumnName\,[\,,ColumnName\,]\,[\,\ldots\,]\,)$$

$$\left\{\,(\,Value\,[\,,Value\,]\,[\,\ldots\,]\,)\,\begin{bmatrix}\text{ROLLBACK WORK}\\\text{COMMIT WORK}\end{bmatrix}\,\right\}[\,\ldots\,]\ \text{E}\left[\text{ND}\right]$$

Parameters

[Owner.]TableName	identifies a table in the DBEnvironment you are using.
[Owner.]ViewName	identifies a view based on a single table. Refer to the discussion of the CREATE VIEW statement in the *ALLBASE/SQL Reference Manual*, for restrictions governing insert operations on a view.
	The new rows are inserted into the table on which the view is based.
ColumnName	identifies the column(s) for which you will specify values. Columns not specified are assigned null values (if allowed).
	You must provide data for any column that does not allow null values.
Value	is data to be inserted into a column. Within each set of parentheses, you specify one row of data; put a space, a carriage return, or a comma between sets of parentheses. A value must be specified for each *ColumnName* specified, in the order in which you named the columns. Provide data in the order in which the columns were named in the table or view definition. Character data that contains a single quote must double the single quote.
	Values for a row can span lines, but do not split values across lines. Enclose character values in single quotation marks. To specify a null value, enter NULL.
ROLLBACK WORK	backs out rows entered since the last COMMIT WORK.
COMMIT WORK	performs a COMMIT WORK. Rows entered up to this point permanently update the DBEnvironment. A COMMIT WORK is also processed if the SET command's AUTOCOMMIT is on every time the number of rows specified in the AUTOSAVE option of the SET command has been entered.

END terminates the INPUT command. If AUTOCOMMIT is on, ISQL issues a COMMIT WORK, and the following message appears:

```
Number of rows processed is n
DBEnvironment has been updated.
```

If AUTOCOMMIT is off, you must explicitly save any unsaved input, as indicated by the following message that appears after you enter END:

```
Number of rows processed is n
COMMIT WORK to save to DBEnvironment.
```

Description

- You must establish a DBE session with the CONNECT or START DBE statement before using the INPUT command.

- ISQL formulates an SQL INSERT statement from the information provided in the INPUT command.

- The INPUT command is processed after you enter END, unless you are in prompting mode. In prompting mode, each row is inserted before you are prompted for the next row.

- In prompting mode, ISQL prompts you for table names, column names, and data values as shown here:

```
isql=> INPUT;

Table name> TableName;
Column names> (ColumnName1, ColumnName2 ... );
1> (Value1, Value2, 'CharValue3');
2>
   ⋮
5> END;
```

- The INPUT command assumes that you are inputting data in the default format for the data type. If you enter something other than a string, or if the string is the wrong size, an error results. An example of a correct INPUT command follows:

```
isql=>INPUT PurchDB.SupplyBatches(VendPartNumber,BatchStamp,
MinPassRate);

1> ('90015', '1984-07-09 11:09:18.432', 0.9732);
2> ('7310',  '1984-07-14 10:11:13.824', 0.9794);
3> END;
Number of rows processed is 2
COMMIT WORK to save to DBEnvironment.

isql=>
```

INPUT

■ Character data that contains a single quote must double the single quote. For example:

```
isql=> INPUT PurchDB.Parts (PartNumber,PartName,SalesPrice)
> ('2002-DA-01', 'Joe''s Plotter', 500.00)
> END;
Number of rows processed is 1
COMMIT WORK to save to DBEnvironment.

isql=>
```

Authorization

You must have authorization to insert a row into the named table or into the table on which the named view is derived. Refer to the discussion of the INSERT statement in the *ALLBASE/SQL Reference Manual*.

Example

In prompting mode:

```
isql=> INPUT PurchDB.Parts (PartNumber);

1> ('2033-KLT-01');
2> ('2034-BLT-01'),('999-99-9999');
4> END;
Number of rows processed is 3
COMMIT WORK to save to DBEnvironment.

isql=> COMMIT WORK;
```

In nonprompting mode:

```
isql=> INPUT PurchDB.SupplyPrice (PartNumber,VendorNumber,VendPartNumber)
> ('2033-KLT-01', 9002, 'A10')
> ('2034-BLT-01', 9003, 'A12')
> END;
Number of rows processed is 2
COMMIT WORK to save to DBEnvironment.

isql=> COMMIT WORK;
```

INSTALL

The INSTALL command stores the modules contained in an installable module file into a DBEnvironment.

Scope

ISQL only.

ISQL Syntax

IN[STALL]*FileName* [DROP][IN *DBEFileSetName*][NO OPTINFO]

Parameters

FileName	identifies the file containing the installable modules. When first created by an ALLBASE/SQL preprocessor, the file is named *SourceFileName*.sqlm. Name qualification follows HP-UX conventions:

      [*PathName*/]*FileName*

      Unless you specify an absolute path name, ISQL assumes that any path name you specify is relative to your current working directory.

DROP	specifies that an existing module with the same name as the module to be installed will be dropped. The default is not to drop an existing module.
IN *DBEFileSetName*	specifies a DBEFileSet where the modules are to be installed.
NO OPTINFO	indicates that the access plan optimization information (generated by SETOPT) is not to be installed into the DBE. The system generated access plan information will be used instead.

Description

■ In prompting mode, ISQL prompts you for an option. For example:

```
isql=> INSTALL;

File name> FileName;
```

■ You must establish a DBE session with the CONNECT or START DBE command before using the INSTALL command.

■ After installation of the module is complete, the following message appears if AUTOCOMMIT is off:

```
Number of sections installed: n
COMMIT WORK to save to DBEnvironment.
```

If AUTOCOMMIT is on, ISQL processes a COMMIT WORK after the module is installed, and the following message appears:

```
        Number of sections installed: n
        DBEnvironment has been updated.
```

- Before installing a module, ALLBASE/SQL determines the validity of each section in the module based on current DBEnvironment objects and authorization. A section is marked invalid, but stored, if it is for a statement that accesses an object that does not exist or that the module owner is not authorized to execute. You can use the VALIDATE statement to revalidate invalid sections or you can let ALLBASE/SQL revalidate sections when you run the program.

- If a module by the same *OwnerName.ModuleName* already exists in the target DBEnvironment, the module is not installed unless the DROP option is specified.

- You can determine the names of the modules in a module installation file by using the ISQL LIST INSTALL command.

- A module stored in the DBEnvironment by a preprocessor, or a module installed in ISQL, should *never* be executed directly from ISQL by typing the name of the module. The module should be executed by running the executable program file.

- A module installation file may contain more than one module. However, the preprocessors insert only a single module into each module installation file. If you want to install several modules from the same file, you must first concatenate the separate module installation files. The following example concatenates MOD1 and MOD2 into MODALL, which can then be used for the installation:

```
    % cat MOD1 MOD2 > MODALL
```

- Refer to one of the ALLBASE/SQL application programming guides for information on preprocessors, modules, and using SQL in application programs.

Authorization

You must have the authority to start a DBE session in the target DBEnvironment.

Example

```
    isql=> INSTALL;

    File name> cex2.sqlm;
    Name of module in this file:  PGMR1.CEX2
    Number of sections installed:  1

    Total number of sections installed:  1
    COMMIT WORK to save to DBEnvironment.

    isql=>
```

LIST FILE

The LIST FILE command displays the contents of a currently accessible file.

Scope

ISQL only.

ISQL Syntax

$$LI \big[ST \big] F \big[ILE \big] FileName$$

Parameters

FileName   identifies the file to be displayed, most likely a command file. Name qualification follows HP-UX conventions:

                                 [PathName/]FileName

                     Unless you specify an absolute path name, ISQL assumes that any path name you specify is relative to your current working directory.

Description

■ In prompting mode, ISQL prompts you for an option as follows:

```
isql=> LIST;

Option (s[et], f[ile], i[nstall], or h[istory])> FILE;
File name> FileName
```

■ As many as 80 bytes per line are listed.

Example

```
isql=> LIST FILE Daily;

---------------------------
|          Daily          |
---------------------------

UPDATE STATISTICS FOR TABLE PurchDB.OrderItems;
UPDATE STATISTICS FOR TABLE PurchDB.Orders;

isql=>
```

LIST HISTORY

The LIST HISTORY command displays one or all commands in the command history buffer.

Scope

ISQL only.

ISQL Syntax

$$\text{LI}\big[\text{ST}\big]\text{H}\big[\text{ISTORY}\big] \left\{ \begin{array}{l} CommandNumber \\ \text{@} \end{array} \right\}$$

Parameters

CommandNumber	is a number in the range 1 through 50, where 1 identifies the most recently submitted ISQL or SQL command in the ISQL session.
@	refers to the entire command history buffer. If @ is the line-kill character on your system, you must precede the @ with the current HP-UX escape character.

Description

■ Display of commands longer than 65 bytes is truncated, and an ellipsis is inserted at the end of the line to indicate truncation.

■ In prompting mode, ISQL prompts you for an option as follows:

```
isql=> LIST;

Option (s[et], f[ile], i[nstall], or h[istory])> HISTORY;
Command number or @> @
```

Example

```
isql=> LIST HISTORY @;

1      SELECT * FROM PurchDB.Parts;
2      SELECT * FROM PurchDB.Vendors;
3      SELECT * FROM PurchDB.PartInfo;
4      CONNECT TO '../sampledb/PartsDBE';
5
6
7
8
9
10

isql=>
```

LIST INSTALL

The LIST INSTALL command displays the names of modules in an installable module file.

Scope

ISQL only.

ISQL Syntax

LI[ST]I[NSTALL]*FileName*

Parameters

FileName   identifies an installable module file. Name qualification follows HP-UX conventions:

[*PathName*/]*FileName*

Unless you specify an absolute path name, ISQL assumes that any path name you specify is relative to your current working directory.

Description

- In prompting mode, ISQL prompts you for a file name as shown here:

```
isql=> LIST;

Option (s[et], f[ile], i[nstall], or h[istory])> INSTALL;
File name> program1.sqlm
```

- You can store the modules in this file in a DBEnvironment by using the ISQL INSTALL command.

Example

```
isql=> LIST INSTALL;
File name> prog1.sqlm;
Name of module in this file:  pgmr1.program1a
Name of module in this file:  pgmr1.program1b

isql=>
```

LIST SET

The LIST SET command displays any or all of the current SET option values.

Scope

ISQL only.

ISQL Syntax

$$\texttt{LI}\big[\texttt{ST}\big]\texttt{S}\big[\texttt{ET}\big]\left\{\begin{array}{l} \textit{Option} \\ \texttt{@} \end{array}\right\}$$

Parameters

Option
names one of the following SET command options:

```
AUTOC[OMMIT]
AUTOL[OCK]
AUTOS[AVE]
C[ONTINUE]
CONV[ERT]
EC[HO]
ECHO_[ALL]
ED[ITOR]
ES[CAPE]
EXIT[_ON_DBERR]
EXIT_ON_DBWARN
FL[AGGER]
F[RACTION]
N[ULL]
LOAD_B[UFFER]
LOAD_E[CHO]
OU[TPUT]
OW[NER]
PA[GEWIDTH]
PR[OMPT]
```

Refer to the SET command for a complete explanation of each option.

@
indicates all SET command options. If @ is the line-kill character on your system, you must precede the @ with the current HP-UX escape character.

Description

■ In prompting mode, ISQL prompts you for an option:

```
isql=> LIST;

Option (s[et], f[ile], i[nstall], or h[istory])> SET;
Valid set options are:
  autoc[ommit]    autol[ock]      autos[ave]      cont[inue]
  conv[ert]       ec[ho]          echo_[all]      ed[itor]
  es[cape]        exit[_on_dberr] exit_on_dbwarn  fl[agger]
  f[raction]      load_b[uffer]   load_e[cho]     n[ull]
  ou[tput]        ow[ner]         pa[gewidth]     pr[ompt]
Set option or @> @;
```

Example

```
isql=> LIST SET ECHO;

ec[ho]              - OFF

isql=> LIST SET @;

autoc[ommit]        - OFF
autol[ock]          - OFF
autos[ave]          - 5
c[ontinue]          - ON
conv[ert]           - OFF
ec[ho]              - OFF
echo_[all]          - OFF
ed[itor]            - /usr/bin/vi
es[cape]            - \
exit[_on_dberr]     - OFF
exit_on_dbwarn      - OFF
fl[agger]           -
f[raction]          - 2
n[ull]              -
load_b[uffer]       - 16384
ou[tput]            - isqlout
ow[ner]             - linda
pa[gewidth]         - 32767
pr[ompt]            - isql=>

isql=>
```

LOAD

The LOAD command inserts rows into a table from a file.

Scope

ISQL only.

ISQL Syntax

$$\texttt{LO[AD][P[ARTIAL]]FROM} \left\{ \begin{array}{l} \texttt{E[XTERNAL]} \\ \texttt{I[NTERNAL]} \end{array} \right\} \textit{InputFileName} \; \texttt{[AT } \textit{StartingRow]}$$

$$\texttt{[FOR } \textit{NumberOfRows}\texttt{]TO} \left\{ \begin{array}{l} \texttt{[Owner.]TableName} \\ \texttt{[Owner.]ViewName} \end{array} \right\} \left[\begin{array}{l} \textit{ExternalInputSpec} \\ \texttt{USING } \textit{DescriptionFileName} \end{array} \right]$$

$$\left\{ \begin{array}{l} \texttt{Y[ES]} \textit{PatternLocation Pattern} \\ \texttt{N[O]} \end{array} \right\}$$

Parameters

PARTIAL	indicates that you want to load only a range of rows from a file.
EXTERNAL	indicates the input file is a text file. The file can be created outside ISQL or with the EXTERNAL option of the ISQL UNLOAD command.
INTERNAL	indicates the input file is one that is created with the INTERNAL option of the ISQL UNLOAD command. The format of this file is recognizable only to ALLBASE/SQL, when executing the LOAD command with the INTERNAL option.
InputFileName	identifies the file containing the rows to be inserted. Name qualification follows HP-UX conventions:

                    [*PathName*/]*FileName*

                    Unless you specify an absolute path name, ISQL assumes that any path name you specify is relative to your current working directory.

StartingRow	identifies the first row to be read from an EXTERNAL file when the PARTIAL option is specified. The default is 1.
NumberOfRows	identifies the total number of rows, beginning with the starting row, to be read from a file. The default is the number of rows from *StartingRow* to the end of the file.
[*Owner*.] *TableName*	identifies the table to be loaded in the DBEnvironment you are using.
[*Owner*.] *ViewName*	identifies a view based on a single table. Refer to the CREATE VIEW command in the *ALLBASE/SQL Reference Manual* for restrictions governing insert operations on a view.

ExternalInputSpec	describes how ISQL should read an EXTERNAL file:

$$\left\{ \begin{array}{l} ColumnName\ StartingLocation\ Length\ [\,NullIndicator\,] \\ [\,FormatType\,] \end{array} \right\}$$

$$[\ \ldots\ ]\,\texttt{E}\,[\,\texttt{ND}\,]$$

ColumnName	identifies the column into which the data at *StartingLocation* is to be inserted. You must provide data for any column that is *not null*.
StartingLocation	is the column (byte position) in each external file line at which the data for *ColumnName* starts. The first column in a line is 1.
Length	is the number of input line columns (bytes) in the data field.
NullIndicator	identifies the character in an input data field that represents the value of null. You must supply a *NullIndicator* for any *ColumnName* that allows null values. The null indicator can be any one-byte character except a blank, a semicolon, a single or double quotation mark, a minus sign, or the current SET ESCAPE character. If you use a slash ($/$), precede it with the current SET ESCAPE character. In the data files, ensure the null indicator is at *StartingLocation*. Also ensure that any data having the null indicator as the first character does not start at *StartingLocation* in the data file.
FormatType	specifies the IBM data type of the column in the external file:

$$\left[\begin{array}{l} \texttt{CHAR} \\ \texttt{INTEGER} \\ \texttt{PACKED}\ Scale \\ \texttt{ZONED}\ Scale \end{array} \right]$$

FormatType is required when the SET CONVERT option is ASCII or EBCDIC, unless the ALLBASE/SQL column being loaded is character or binary. The *Scale* parameter is an integer indicating the number of digits to the right of the decimal point.

END	indicates end of column descriptions.

DescriptionFileName   identifies a description file. Name qualification follows the same conventions used for the output file.

The description file contains one line for each column in the table. The first line describes the first column in the table, the second line describes the second column in the table, and so on. Each line contains seven fields. Data in all the fields except the column name field are right-justified; the column name field is left justified.

Column(s)	Contents
1-20	Column name.

25	LOAD/UNLOAD type code (see following list):
	
	 0 = SMALLINT & INTEGER
	 1 = BINARY
	 2 = CHAR
	  DATE[1]
	  TIME[1]
	  DATETIME[1]
	  INTERVAL[1]
	 3 = VARCHAR
	 4 = FLOAT
	 5 = DECIMAL
	 8 = NATIVE CHAR[2]
	 9 = NATIVE VARCHAR[2]
	14 = VARBINARY
	15 = LONG BINARY
	16 = LONG VARBINARY
	
	[1] DATE, TIME, DATETIME and INTERVAL columns are known to the LOAD and UNLOAD commands as CHAR data.
	
	[2] NATIVE CHAR or VARCHAR is what SQLCore uses internally when a CHAR or VARCHAR column is defined with a LANG= clause.

30-40	Output length.
41-50	Fraction length (number of decimal places).
51-60	Starting location of data.
61-70	Starting location of length of VARCHAR data.
75	Null indicator.

A description file is generated automatically when using the UNLOAD EXTERNAL command, but can be created with an editor. When SET CONVERT is ASCII or EBCDIC, a description file cannot be specified.

YES   indicates that you want to load only rows that meet specific criteria.

PatternLocation   identifies the input line column at which a *Pattern* starts.

Pattern   is a character string that constitutes the criteria for loading. Only rows that have the *Pattern* starting at *PatternLocation* are loaded.

NO   indicates that you want to load all the rows in the external file.

Description

■ You must establish a DBE session with the CONNECT or the START DBE command before using the LOAD command.

■ If a tape is being used, mount the tape and issue the LOAD command at the ISQL prompt; otherwise just issue the LOAD command. When the end of the tape is reached, the ISQL prompt reappears. To continue loading data from a second tape, mount the tape and reissue the LOAD command. You can use the REDO command to enter the same syntax again.

■ When loading has begun, the following message is displayed.

```
Command in progress
```

■ During table loading from either external or internal files, ISQL displays the cumulative number of rows read and rows loaded after it processes a group of rows. For example:

```
Number of rows read is 12
Number of rows processed is 12
Number of rows read is 24
Number of rows processed is 24

    ⋮
```

The number of rows in each group inserted depends on the columns and their sizes.

■ If AUTOCOMMIT is off, the following message appears after loading is complete:

```
Number of rows read is n
Number of rows processed is n
COMMIT WORK to save to DBEnvironment.
```

■ If AUTOCOMMIT is on, ISQL will continue to load rows until the 16K tuple buffer is filled, even if the number of rows specified in the AUTOSAVE option has been exceeded. Once the 16K tuple buffer is full, a check is made to see if the number of rows specified in the AUTOSAVE option has been reached or exceeded. If this is the case, ISQL processes a COMMIT WORK statement. For more information on the 16K tuple buffer, refer to the "Concurrency Control through Locks and Isolation Levels" chapter in the *ALLBASE/SQL Reference Manual*. A COMMIT WORK is also processed after the last row is loaded; then the following message appears:

```
Number of rows read is n
Number of rows processed is n
DBEnvironment has been updated.
```

LOAD

- To improve performance, issue the following SET commands prior to the load operation:

  □ `SET LOAD_BUFFER` to enlarge the load buffer beyond the default size of 16,384 bytes.

  □ `SET AUTOLOCK ON` to lock the table in exclusive mode when the load is performed.

  □ `SET AUTOCOMMIT ON` and `SET AUTOSAVE` to automatically commit the number of rows specified by autosave if the load buffer is full. Should the load operation subsequently fail, you can insert the remaining rows with the LOAD PARTIAL command.

  □ `SET SESSION DML ATOMICITY AT ROW LEVEL` to reduce logging overhead when running in non-archive mode.

  □ `SET SESSION UNIQUE, REFERENTIAL, CHECK CONSTRAINTS DEFERRED` to defer constraint checking until the end of the load operation.

  For more hints on improving load performance, refer to the "Initial Table Loads" section in the *ALLBASE/SQL Performance and Monitoring Guidelines*

- If the LOAD command is terminated due to an error, you can correct the problem, then insert the remaining rows by submitting a LOAD PARTIAL command after the last row read. However, if the error causes your transaction to be terminated, any rows loaded since the last COMMIT WORK are rolled back. In this case, submit a LOAD PARTIAL command after the last row read before the last COMMIT WORK.

- In prompting mode, ISQL prompts you for input options as follows:

  ```
  isql=> LOAD PARTIAL;

  File format (i[nternal] or e[xternal])> INTERNAL;
  Input file name> InputFileName;
  Starting Row> StartingRow;
  Number of Rows> NumberOfRows;
  Table name> TableName;
  ```

  You are prompted for *StartingRow* and *NumberOfRows* only if the PARTIAL option is specified.

- When you are loading from an external file in prompting mode, ISQL prompts you for *DescriptionFileName* if you specified the keyword USING. If the keyword USING was not specified, ISQL prompts you for a description of each input column. For example:

  ```
  Enter information as requested for each column to be loaded.
  Enter END for the column name when finished.
  Column name> ColumnName;
  Starting location> StartingLocation;
  Column length> Length;
  Null representation> NullRepresentation;
  Column name>
  ```
            ⋮

  You are prompted for a *NullIndicator* only if the column you are describing allows null values.

■ When loading from an external file created by using the UNLOAD command, specify a *StartingLocation* after the 10-byte prefix if the file contains VARCHAR data. The 10-byte prefix, written by the UNLOAD command using the EXTERNAL option, contains the actual length of the VARCHAR data item.

■ If a column can contain null values but the null indicator you specify is not in the proper column in an external data file, ISQL loads space(s) into character columns, loads zero(s) into numeric columns, and generates an error message for date/time columns.

■ If a column is not null and the external data file contains spaces, ISQL loads space(s) into character columns, loads zero(s) into numeric columns, and generates an error message for date/time columns.

■ When loading a large number of rows, use the LOCK TABLE statement in EXCLUSIVE mode before using the LOAD command. This keeps the load process from obtaining a large number of page locks and thus depleting shared memory for the lock table. Alternatively, you can increase the number of control block pages by using the SQLUtil ALTDBE command.

■ If you create a description file with an editor, the length of a BINARY column is double the actual length of the column. This is caused by the description file being in ASCII format where one character is translated to a byte; however, in hexadecimal format two bytes make one character. So for ALLBASE/SQL to read the correct number of bytes, the actual length is doubled.

■ When loading a LONG column and you are prompted for the output length, respond with the length of the output file name, not the length of the data type.

■ In external files, DATE, TIME, DATETIME, and INTERVAL columns appear as characters. Internally, they are stored as binary values, although ISQL returns a code of 2 (CHAR) for them in creating description files.

■ If the SET CONVERT option is ASCII or EBCDIC, ISQL converts the IBM mainframe data from the *InputFileName* file during the LOAD operation. The supported conversions of IBM mainframe data when loading ALLBASE/SQL columns are as follows:

Valid IBM Mainframe Format Types

Target ALLBASE/SQL Column	IBM Mainframe Format Type	ExternalInputSpec FormatType
INTEGER or SMALLINT	INTEGER PACKED DECIMAL ZONED DECIMAL[1] ASCII EBCDIC	INTEGER PACKED *Scale* ZONED *Scale* CHAR CHAR
DECIMAL	PACKED DECIMAL ZONED DECIMAL[1] ASCII EBCDIC	PACKED *Scale* ZONED *Scale* CHAR CHAR
CHAR or VARCHAR	ASCII EBCDIC	none none
BINARY or VARBINARY	BINARY	none

[1] A ZONED DECIMAL number may have either a leading or trailing sign.

The EXTERNAL parameter must be specified. The *DescriptionFileName* parameter is not allowed. Since NULL values are not permitted, ISQL does not prompt for them. ISQL generates additional prompts for the format type of the *InputFileName* data only if the data type of the ALLBASE/SQL column is INTEGER, SMALLINT, or DECIMAL.

If the format type to be loaded is PACKED DECIMAL or ZONED DECIMAL, ISQL prompts for the *Scale*, which is the number of digits to the right of the decimal point. The location of the decimal point is determined by the response to the *Scale* prompt, not by the column definition. When loading an ALLBASE/SQL INTEGER or SMALLINT column with PACKED DECIMAL or ZONED DECIMAL data, the digits to the right of the decimal point are truncated. If an ALLBASE/SQL numeric column is not large enough to hold the incoming PACKED DECIMAL or ZONED DECIMAL data, an error occurs.

Authorization

You must be authorized to insert a row into the table named. Refer to the INSERT command in the *ALLBASE/SQL Reference Manual*.

Example

A COMMIT WORK is automatically performed when 40 rows have been loaded from the external Price file into the PurchDB.SupplyPrice table. ISQL prompts for the column name, starting location, column length, and null representation.

```
isql=> SET AUTOCOMMIT ON;
isql=> SET AUTOSAVE 40;
isql=> LOAD FROM EXTERNAL Price TO PurchDB.SupplyPrice;

Enter information as requested for each column to be loaded.
Enter END for the column name when finished.
Column name> PartNumber;
Starting location> 1;
Column length> 16;
Column name> VendorNumber;
Starting location> 17;
Column length> 4;
Column name> VendPartNumber;
Starting location> 24;
Column length> 6;
Column name> UnitPrice;
Starting location> 33;
Column length> 7;
Null representation> ?;
Column name> DeliveryDays;
Starting location> 42;
Column length> 3;
Null representation>  ?;
Column name> END;
Load depending on value in input record (Y/N)> Y;
Starting location of select field> 17;
Select field pattern> 9014;
Command in progress
Number of rows read is 25
Number of rows processed is 15
Number of rows read is 55
Number of rows processed is 30
Number of rows read is 75
Number of rows processed is 42
DBEnvironment has been updated.
```

*A COMMIT WORK is automatically performed when 50 rows have been loaded from the
external EParts file into the PurchDB.Parts table. ISQL does not prompt for the external input
specification values, since they are included as parameters of the LOAD command.*

LOAD

```
isql=> SET AUTOSAVE 50
isql=> LOAD FROM EXTERNAL EParts
> TO PurchDB.Parts
> PartNumber 1      16
> PartName    17    30      ?
> SalesPrice 47      7      ?
> END
> N;
>
Command in progress
Number of rows read is 25
Number of rows processed is 25
Number of rows read is 50
Number of rows processed is 50
DBEnvironment has been updated.
Number of rows read is 75
Number of rows processed is 75
Number of rows read is 78
Number of rows processed is 78
DBEnvironment has been updated.

isql=>
```

Starting with row 5, load 60 rows from the external file ExtParts into the PurchDB.Parts table.

```
isql=> LOAD PARTIAL FROM EXTERNAL ExtParts
> AT 5
> FOR 60
> TO PurchDB.Parts
> USING DesParts;
Command in progress
Number of rows read is 25
Number of rows processed is 25
Number of rows read is 50
Number of rows processed is 50
DBEnvironment has been updated.
Number of rows read is 60
Number of rows processed is 60
DBEnvironment has been updated.

isql=>
```

Load from the external file Ecustomer, which was created on an IBM mainframe, into the Customer table. The character data in the external file is in EBCDIC format.

```
isql=>INFO Customer;

Column Name              Data Type (length)     Nulls Allowed     Language
--------------------------------------------------------------------------
CUSTOMERID               Char     (  10)            NO             n-computer
CREDITAMOUNT             Decimal  (  10,    4)      NO
QTYSOLD                  SmallInt                   NO
TESTDATA                 Binary   (  20)            NO

isql=> SET CONVERT EBCDIC;

isql=> LOAD FROM EXTERNAL ECustomer TO Customer
> CustomerId     1   8
> CreditAmount   9   4 packed 2
> QtySold        13  4 integer
> TestData       17  8
> END
> N;
>
Command in progress.
Number of rows read is 64
Number of rows processed is 64
COMMIT WORK to save to DBEnvironment.

isql=>
```

RECALL

The RECALL command displays the current contents of the command buffer (up to 32K bytes). Optionally, it places a command from the command history buffer or the contents of a file into the command buffer and displays it.

Scope

ISQL only.

ISQL Syntax

$$\text{REC}\left[\text{ALL}\right]\left\{\begin{array}{l} \text{C}\left[\text{URRENT}\right] \\ \text{F}\left[\text{ILE}\right]\textit{FileName} \\ \text{H}\left[\text{ISTORY}\right]\textit{CommandNumber} \end{array}\right\}$$

Parameters

CURRENT
: displays the current contents of the command buffer (including comments). The following commands place information into the command buffer:

    HOLD
    RECALL FILE
    RECALL HISTORY

FILE *FileName*
: puts the contents of a file into the command buffer and displays it (including comments). File name qualification follows HP-UX conventions:

    [*PathName*/] *FileName*

    Unless you specify an absolute path name, ISQL assumes that any path name you specify is relative to your current working directory.

HISTORY *CommandNumber*
: puts one of the commands currently in the command history buffer into the command buffer and displays it. The command history buffer holds the ten most recently submitted commands, numbered 1 through 10. The *CommandNumber* of the most recently submitted command is 1. You can use the LIST HISTORY @ command to identify the *CommandNumber* associated with each command currently in the command history buffer.

Description

■ All options of the RECALL command concatenate command lines. For example:

```
isql=> RECALL FILE FileName;

UPDATE STATISTICS FOR TABLE PurchDB.OrderItems; ...; UP
DATE STATISTICS FOR TABLE PurchDB.Inventory;

isql=>
```

■ If RECALL HISTORY 1 is invoked right after a multiple-command command file is executed, the command displayed is the last SQL or ISQL command processed, not all the commands in the command file.

■ When commands are in the command buffer, you can operate on them with the following ISQL commands:

```
CHANGE
RECALL CURRENT
START
STORE
```

■ In prompting mode, ISQL prompts you for RECALL options as follows:

```
isql=> RECALL;

Recall option (c[urrent], f[ile], or h[istory])> HISTORY;
Command number> 1;
COMMIT WORK;

isql=>
```

■ Up to 32K bytes of a command file are stored in the command buffer and displayed.

Example

```
isql=> HOLD SELECT * FROM
> PurchDB.Orders;
isql=> RECALL CURRENT;

SELECT * FROM PurchDB.Orders;

isql=> SET ECHO OFF;
isql=> START Daily;
isql=> RECALL HISTORY 1;

UPDATE STATISTICS FOR TABLE PurchDB.Orders;

isql=>
```

REDO

The REDO command lets you edit any of the commands in the command history buffer and optionally execute the edited command. The original command in the command history buffer is left unchanged.

Scope

ISQL only.

ISQL Syntax

$$\text{RED}\begin{bmatrix} 0 \end{bmatrix}\begin{bmatrix} CommandNumber \\ CommandString \end{bmatrix}$$

Parameters

CommandNumber   identifies one of the commands in the command history buffer. The command history buffer holds the fifty most recently submitted commands, numbered 1 through 50. The *CommandNumber* of the most recently submitted command is 1.

CommandString   identifies the most recent command in the command history buffer that begins with the *CommandString*.

Description

■ You can use the LIST HISTORY @ command to identify the *CommandNumber* associated with each command currently in the command history buffer.

■ If no parameter is specified, the most recently submitted command is assumed.

■ REDO cannot be used from a command file or the command buffer.

■ Once the REDO command is submitted, you are placed in edit mode, and the first line of the command is displayed. You enter the following editing subcommands on the line beneath the displayed line, referred to as the **current edit line**. You can use either uppercase or lowercase for the alphabetic subcommands listed here:

      B   breaks the line being edited by moving the character above the B and all subsequent characters on the line to the next line, which becomes the current edit line.

      D   deletes the character above the D. You can delete multiple characters by entering the D below the first and the last character to be deleted, leaving blanks between them. You can delete, then insert, characters by using one or more Ds followed by an I.

      E   exits the REDO command without executing the edited command.

      H   lists all the valid editing subcommands, then redisplays your current edit line.

I   inserts one or more characters starting at the location of the I. You can delete, then insert, characters by using one or more Ds followed by an I.

L   lists the complete command as it is currently edited, then redisplays the current edit line.

R   replaces one or more characters in the line being edited. Characters are replaced beginning with the character above the R one for one with any characters you enter following the R. The R can be omitted unless one of the other edit subcommands is the first replacement character.

X   executes the edited version of the command.

+[n]   makes the nth line forward in the command being edited available for editing; n is 1 by default. If you enter a number greater than the number of lines remaining in the command, the last line in the command is made available for editing.

-[n]   makes the nth line back in the command being edited available for editing; n is 1 by default. If you enter a number greater than the number of lines earlier in the command, the first line in the command is made available for editing.

(Return)   makes the next line of the command being edited available for editing. If the current edit line is the last line, the edited command is executed.

- Any characters other than those listed above are interpreted as replacement characters; they replace the characters above them in the current edit line.

- Use the space bar, not the arrow keys to move forward on the line to be edited.

- The edit line cannot exceed 72 characters. If inserting additional characters will cause the line to exceed 72 characters, use the B subcommand to break the line before inserting.

Example

```
isql=> SELECT PartName, SalesPrice
> FROM PurchDB.Prts
> WHERE PartName IS NUL;

SELECT PartName, SalesPrice FROM PurchDB.Prts WHERE PartName IS NUL;
                                                                   |
Syntax error.  (DBERR 1001)
isql=> REDO;
SELECT PartName, SalesPrice FROM PurchDB.Prts WHERE PartName IS NUL;
                  B
SalesPrice FROM PurchDB.Prts WHERE PartName IS NUL;
-1
SELECT PartName,
                IPartNumber,
SELECT PartName, PartNumber,
+
SalesPrice FROM PurchDB.Prts WHERE PartName IS NUL;
                                a
SalesPrice FROM PurchDB.Pats WHERE PartName IS NUL;
H
Valid commands are:  B, D, E, H, I, L, R, X, +n, -n
SalesPrice FROM PurchDB.Pats WHERE PartName IS NUL;
                                ir
SalesPrice FROM PurchDB.Parts WHERE PartName IS NUL;
                                                    RL;
SalesPrice FROM PurchDB.Parts WHERE PartName IS NULL;
L

 1. SELECT PartName, PartNumber,
 2. SalesPrice FROM PurchDB.Parts WHERE PartName IS NULL;

SalesPrice FROM PurchDB.Parts WHERE PartName IS NULL;
[ Return ]
```

Edited command is executed.

```
isql=>

isql=> REDO SEL;
```

*The most recent command in the command buffer beginning with the string
"SEL" is displayed in edit mode.*

```
SELECT PartName, PartNumber, SalesPrice FROM PurchDB.Parts WHERE
```

```
PartName IS NULL;
```
[Return]

The command is executed.

RENAME

The RENAME command changes the name of a file.

Scope

ISQL only.

ISQL Syntax

REN[AME] *OldFileName NewFileName*

Parameters

OldFileName   identifies the file to be renamed. Name qualification follows HP-UX conventions:

         [*PathName/*]*FileName*

         Unless you specify an absolute path name, ISQL assumes that any path name you specify is relative to your current working directory.

NewFileName   is the new name. File name qualification follows the conventions given for *OldFileName*.

Description

In prompting mode, ISQL prompts you for the old and new command file names:

```
isql=> RENAME;
Existing file name> OldFileName;
New name> NewFileName;
isql=>
```

Example

```
isql=> RENAME Daily Stats;
isql=>
```

SELECTSTATEMENT

When you enter the SQL SELECT statement through ISQL, the query result is displayed. You can scroll through the query result at the terminal and optionally print it.

Scope

ISQL only.

ISQL Syntax

$SelectStatement;\,[\,\texttt{PA}\,[\,\texttt{USE}\,]\,;\,]\,[\,BrowseOption;\,]\,[\,\ldots\,]\,\texttt{E}\,[\,\texttt{ND}\,]$

Parameters

SelectStatement	is an SQL SELECT statement. Refer to the *ALLBASE/SQL Reference Manual* for additional information on the SELECT statement.
PAUSE	is specified in a command file or command buffer to suspend processing of commands after a SELECT statement is executed. Example:

```
SELECT * FROM PurchDB.Parts;
PAUSE;
```

The query result is displayed and can be examined using a *BrowseOption*. When you enter END on the display, control returns to the command file or the command buffer.

If you do not use the PAUSE option, you must specify END.

BrowseOption is specified in a command file or in response to the prompt beneath the query result display. Valid options are listed here:

$$\left\{ \begin{array}{l} \texttt{U}\,[\,\texttt{P}\,] \\ \texttt{D}\,[\,\texttt{OWN}\,] \\ \texttt{L}\,[\,\texttt{EFT}\,] \\ \texttt{R}\,[\,\texttt{IGHT}\,] \\ \texttt{T}\,[\,\texttt{OP}\,] \\ \texttt{B}\,[\,\texttt{OTTOM}\,] \\ \texttt{N}\,[\,\texttt{EXT}\,] \\ \texttt{PR}\,[\,\texttt{INT}\,]\,[\,NumberOfCopies\,] \end{array} \right\}$$

U[P]   scrolls backward by half the number of displayed rows, along with the previous rows, and preserves the column headings.

	D[OWN]	scrolls forward by half the number of displayed rows, along with the previous rows, and preserves the column headings. (Pressing ⌈Return⌉ has the same effect as pressing D.)
	L[EFT]	scrolls the display left 40 screen columns.
	R[IGHT]	scrolls the display right 40 screen columns.
	T[OP]	displays the first group of rows and the column headings.
	B[OTTOM]	displays the last group of rows and the column headings.
	N[EXT]	displays the next group of rows and the column headings.
	PR[INT] *NumberOfCopies*	copies the query result to the device specified in the environment variable *lpdest*. If *lpdest* has not been set, the output goes to the system default destination. *NumberOfCopies* can be from 1 (the default) through 20.
E[ND]		terminates the query result display. If the PAUSE option is in effect, control returns to the command file. If the OUTPUT option of the SET command is set to a file other than *isqlout*, the entire query result is written to that file. If, however, you enter a slash (/) to terminate the query result display, only the rows displayed so far are written to the output file.

Description

- The number of screen columns displayed is dependent on the screen size of your terminal.

- A command entry prompt appears at the bottom of the display:

U[p], d[own], l[eft], r[ight], t[op], b[ottom], n[ext], pr[int] <n> , or e[nd]>

You enter browse options from this list to manipulate the display.

- The number of rows already displayed appears beneath the query result as follows:

    First n rows have been selected.

The number of rows selected is incremented whenever you enter DOWN. The total number of rows in the query result is displayed when you enter BOTTOM, or when the last row in the query result has been displayed:

    Number of rows selected is n.

- The maximum length of a row in a query result obtained through ISQL is the PAGEWIDTH option value of the SET command, which can be as large as 32767 bytes. If a row is longer than the PAGEWIDTH option value, then the following occurs:

  □ Column headings are truncated.

  □ CHAR or VARCHAR data is truncated after the PAGEWIDTH column.

    □ Numeric data is not displayed or printed unless all the data fits.

- The following SET command options also affect the query result:

FRACTION
: determines the number of decimal digits displayed for FLOAT and DECIMAL data.

NULL
: determines the character ISQL displays to signify a null value.

OUTPUT
: identifies the file to which the query result is sent. The query result is stored in this file, as well as displayed at the terminal. Each line in the file is as wide as the current PAGEWIDTH option value of the SET command. If the file is isqlout, it is removed when the END option is processed. If the file is named something other than isqlout, it is created. If the output file already exists, it is overwritten with the new output. You can access and manipulate the file with an editor.

- Object names enclosed in double quotes cannot be split across lines.

- When you execute the SELECT statement on a LONG column, the output device is displayed, not data. The output device will contain the actual data which was selected.

- When you execute the SELECT statement on a BINARY or VARBINARY column, the hexadecimal representation is displayed.

Authorization

To issue the SELECT statement, you must have the authority defined in the *ALLBASE/SQL Reference Manual*.

SELECTSTATEMENT

Example

```
isql=> SELECT * FROM System.Table;

SELECT * from System.Table;
-------------------+-------------------+-------------------+------+-----
NAME               |OWNER              |DBEFILESET         |TYPE  |RTYPE
-------------------+-------------------+-------------------+------+-----
COUNTER            |SYSTEM             |SYSTEM             |    0|
USER               |SYSTEM             |SYSTEM             |    0|
TRANSACTION        |SYSTEM             |SYSTEM             |    0|
CALL               |SYSTEM             |SYSTEM             |    0|
ACCOUNT            |SYSTEM             |SYSTEM             |    0|
TABLE              |SYSTEM             |SYSTEM             |    1|
COLUMN             |SYSTEM             |SYSTEM             |    1|
INDEX              |SYSTEM             |SYSTEM             |    1|
SECTION            |SYSTEM             |SYSTEM             |    1|
DBEFILESET         |SYSTEM             |SYSTEM             |    1|
DBEFILE            |SYSTEM             |SYSTEM             |    1|
SPECAUTH           |SYSTEM             |SYSTEM             |    1|
TABAUTH            |SYSTEM             |SYSTEM             |    1|
COLAUTH            |SYSTEM             |SYSTEM             |    1|
MODAUTH            |SYSTEM             |SYSTEM             |    1|
GROUP              |SYSTEM             |SYSTEM             |    1|
----------------------------------------------------------------------------
First 16 rows have been selected.
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd]>r;


-+-------------------+------+------+----------+-----------+-----------+---
 |DBEFILESET         |TYPE  |RTYPE |NUMC      |NUMI       |NUMIC      |NR
-+-------------------+------+------+----------+-----------+-----------+---
 |SYSTEM             |    0|    3|        3|         0|         0|
 |SYSTEM             |    0|    3|        2|         0|         0|
 |SYSTEM             |    0|    3|        4|         0|         0|
 |SYSTEM             |    0|    3|        5|         0|         0|
 |SYSTEM             |    0|    3|        6|         0|         0|
 |SYSTEM             |    1|    0|       12|         0|         2|
 |SYSTEM             |    1|    0|       10|         0|         0|
 |SYSTEM             |    1|    0|       10|         0|         0|
 |SYSTEM             |    1|    0|        7|         0|         0|
 |SYSTEM             |    1|    0|        5|         0|         0|
 |SYSTEM             |    1|    0|        7|         0|         0|
 |SYSTEM             |    1|    0|        4|         0|         0|
 |SYSTEM             |    1|    0|       10|         0|         0|
 |SYSTEM             |    1|    0|        4|         0|         0|
 |SYSTEM             |    1|    0|        3|         0|         0|
 |SYSTEM             |    1|    0|        4|         0|         0|
----------------------------------------------------------------------------
First 16 rows have been selected.
U[p], d[own], l[eft], r[ight], t[op], b[ottom], pr[int] <n>, or e[nd]>e;
```

SET

The SET command defines a number of ISQL environment options.

Scope

ISQL only.

ISQL Syntax

SE[T] *Option OptionValue*

Parameters

Option OptionValue	identifies the option and a value for it. The available options and values are listed here:
AUTOC[OMMIT] ON	Initial setting is OFF.
	If ON, a COMMIT WORK statement is automatically performed: 1) when you are using the INPUT or LOAD command and you enter the number of rows defined in the AUTOSAVE option, 2) when you enter END to terminate the INPUT command, or 3) after a module has been installed with the INSTALL command.
AUTOC[OMMIT] OFF	If OFF, rows inserted with the INPUT command are committed only when you enter COMMIT WORK. Rows inserted with the LOAD command or modules installed with the INSTALL command are committed only when you enter COMMIT WORK.
AUTOL[OCK] ON	Initial setting is OFF.
	If ON, ISQL automatically acquires an exclusive lock upon the table when the LOAD command is executed. Setting AUTOLOCK to ON improves performance when large data files are loaded.
AUTOL[OCK] OFF	The LOAD command does not automatically lock the table.
AUTOS[AVE] *NumberOfRows*	Initial setting is 5.
	If AUTOCOMMIT is ON, ISQL will continue to load rows with the INPUT or LOAD command until the load buffer is full, even if the *NumberOfRows* specified has been exceeded. Once the load buffer is full, a check is made to see if the *NumberOfRows* value has been reached or exceeded. If this is the case ISQL processes a COMMIT WORK statement.
	NumberOfRows can be from 0 through 2,147,483,647. When using the LOAD command for large data files, you should increase *NumberOfRows* above the initial setting of 5. For the INPUT command, *NumberOfRows* should not be greater than 300.

SET

C[ONTINUE] ON	Initial setting is ON.
	If ON, ISQL continues processing commands from a command file after encountering an error.
C[ONTINUE] OFF	If OFF, ISQL terminates command file processing after encountering an error.
CONV[ERT] ASCII	If ASCII, the LOAD command converts the data in an external file from IBM mainframe format to an ALLBASE/SQL data type. Character data in the external file is in ASCII format.
CONV[ERT] EBCDIC	If EBCDIC, the LOAD command converts the data in an external file from IBM mainframe format to an ALLBASE/SQL data type. Character data in the external file is in EBCDIC format.
CONV[ERT] OFF	Initial setting is OFF. If OFF, the LOAD command will not convert the data in an external file.
EC[HO] ON	If ON, commands and comments in command files are displayed as they are read from the command file.
EC[HO] OFF	Initial setting is OFF.
	If OFF, commands and comments read from command files are not displayed.
ECHO_[ALL] ON	If ON, batch user input is echoed to standard out.
ECHO_[ALL] OFF	Initial setting is OFF.
	If OFF, batch user input is not echoed.
ED[ITOR] *EditorName*	Initial setting is /usr/bin/vi.
	The named editor is invoked when you execute the EDIT command. If you do not specify a full path name, ISQL uses the search paths defined in your .profile or .cshrc file.
ES[CAPE] *Character*	Initial setting is a backslash (\).
	Characters with special meaning to ISQL must be preceded by the one-byte escape character when used within an ISQL command.
EXIT[_ON_DBERR] ON	If ON, ISQL terminates immediately if an SQL error is encountered.
EXIT[_ON_DBERR] OFF	Initial setting is OFF.
	If OFF, ISQL does not immediately terminate if an SQL error is encountered.
EXIT_ON_DBWARN ON	If ON, ISQL terminates immediately if an SQL warning is encountered.
EXIT_ON_DBWARN OFF	Initial setting is OFF.
	If OFF, ISQL does not immediately terminate if an SQL warning is encountered.

FL[AGGER] [*StandardName*]	Initial setting is blank, indicating that SQL statements are not checked for standards compliance.
	ISQL displays a warning for each SQL statement that does not conform to the *StandardName* specified. The *StandardName* option is not case sensitive. To turn off the flagger, issue this command without the *StandardName* option. See the "Standards Flagging Support" appendix in the *ALLBASE/SQL Reference Manual* for more information on ALLBASE/SQL standards compliance.
F[RACTION] *Length*	Initial setting is 2.
	The *length* determines the number of digits displayed to the right of the decimal point in the output of the SELECT statement for a column containing data of type DECIMAL, FLOAT, or REAL. *Length* can be from 0 through 18.
LOAD_B[UFFER] *NumberOfBytes*	Initial setting is 16,384.
	The *NumberOfBytes* specifies the size of the buffer ISQL uses with the LOAD and UNLOAD commands.
LOAD_E[CHO] ON	Initial setting is ON.
	If ON, ISQL reports the number of rows processed by each BULK INSERT executed by a LOAD command or each FETCH issued by an UNLOAD command. If AUTOCOMMIT is ON, ISQL displays a message each time a COMMIT WORK is automatically performed by the LOAD command.
LOAD_E[CHO] OFF	If OFF, ISQL displays status messages for the LOAD and UNLOAD commands only after all rows have been processed.
LOAD_E[CHO] AT_COMMIT	If AT_COMMIT, ISQL only displays status messages when a COMMIT WORK is issued by a LOAD command.
N[ULL] [*Character*]	Initial setting is blank.
	This one-byte character is displayed when a null value occurs in a column. The NULL character can be any one-byte character, except a semicolon or a comma. To use a single or double quotation mark or the current SET ESCAPE character, precede it with the current SET ESCAPE character.
OU[TPUT] *FileName*	Initial setting is isqlout.
	All SELECT results go to the named file. ISQL creates the file when you execute the SELECT statement. If it already exists, it is overwritten. If the file is named isqlout, it is removed when the query result display is terminated. Each line in the output file is as wide as the PAGEWIDTH setting.
	If the output file contains sensitive data, verify that the file's permissions prohibit access by unauthorized users. The permissions of the output file are affected by the user's file creation mode mask. See the umask(2) and chmod(2) man pages for more information.

`OW[NER]` *OwnerName*	Initial setting is user's DBEUserID name.
	This command temporarily resets the default owner name. It does not change the ownership of existing objects in the DBEnvironment nor does it change any authorities in the DBEnvironment. The new owner name must be 20 bytes or fewer in length. Setting the owner name eliminates the need to fully qualify table and view names in SQL statements. To reset the owner name to the login name, type `SET OWNER` with no owner name.
`PA[GEWIDTH]` *PageWidth*	Initial setting is 254.
	The maximum number of bytes per row of data for SELECT output is the *PageWidth* value. *PageWidth* can be as large as 32767. Blanks pad the line to the option value.
`PR[OMPT]` *PromptString*	Initial setting is `isql=>` .
	The *PromptString* appears when ISQL is ready to accept a command from the terminal. The *PromptString* can be as long as 16 bytes. If the string contains any blanks or commas, enclose the string in single or double ASCII quotes.

Description

■ The EXIT_ON_DBERR option has precedence over the CONTINUE option. For example, if EXIT_ON_DBERR is ON and CONTINUE is ON, ISQL terminates immediately if an SQL error is encountered.

■ The SET command described in this section is defines only ISQL environment options. It is unrelated to the SQL SET statements documented in the *ALLBASE/SQL Reference Manual*.

■ In prompting mode, ISQL displays the following options:

```
isql=> SET;
Valid set options are:
  autoc[ommit]    autol[ock]      autos[ave]      cont[inue]
  conv[ert]       ec[ho]          echo_[all]      ed[itor]
  es[cape]        exit[_on_dberr] exit_on_dbwarn  fl[agger]
  f[raction]      load_b[uffer]   load_e[cho]     n[ull]
  ou[tput]        ow[ner]         pa[gewidth]     pr[ompt]
Set option>
```

Example

```
isql=> LIST SET @;

autoc[ommit]         - OFF
autol[ock]           - OFF
autos[ave]           - 5
c[ontinue]           - ON
conv[ert]            - OFF
ec[ho]               - OFF
echo_[all]           - OFF
ed[itor]             - /usr/bin/vi
es[cape]             - \
exit[_on_dberr]      - OFF
exit_on_dbwarn       - OFF
fl[agger]            -
f[raction]           - 2
n[ull]               -
ou[tput]             - isqlout
ow[ner]              - linda
load_b[uffer]        - 16384
pa[gewidth]          - 32767
pr[ompt]             - isql=>

isql=> SET EDITOR vi;
isql=>
```

SQLGEN

The SQLGEN command invokes SQLGEN.

Scope

ISQL only.

ISQL Syntax

SQLG[EN]

Description

■ Once you submit the SQLGEN command, you can submit only SQLGEN commands until you enter EXIT.

■ SQLGEN uses the editor specified by the SET EDITOR setting.

■ Refer to the *ALLBASE/SQL Database Administration Guide* for information on SQLGEN commands.

Example

```
isql=> SQLGEN;


                              MON JAN 20  12:33:05 1992
HP36217-02A.F0.00 SQL Command Generator   ALLBASE/SQL
(C)COPYRIGHT HEWLETT-PACKARD CO.1986,1987,1988,1989,
1990,1991,1992. ALL RIGHTS RESERVED.


Current Editor: /usr/bin/vi
```

When the SQLGEN prompt (>>) appears, you can submit SQLGEN commands.
To return to ISQL, enter EXIT.

```
>> EXIT

isql=>
```

SQLUTIL

The SQLUTIL command invokes SQLUtil.

Scope

ISQL only.

ISQL Syntax

SQLU[TIL]

Description

■ Once you submit the SQLUTIL command, you can submit only SQLUtil commands until you enter EXIT or QUIT.

■ Refer to the *ALLBASE/SQL Database Administration Guide* for information on SQLUtil commands.

Example

```
isql=> SQLUTIL;


                                      MON,  JAN 20  9:44:18 1992
HP36217-02A.F0.00           DBE Utility/9000   ALLBASE/SQL
(C)COPYRIGHT HEWLETT-PACKARD CO. 1982,1983,1984,1985,1986,1987,
1988,1989,1990,1991,1992.  ALL RIGHTS RESERVED.


>> SQLUTILcommand
```

When the SQLUtil prompt (>>) appears, you can submit SQLUtil commands. To return to ISQL, enter EXIT or QUIT.

```
>> EXIT

isql=>
```

START

The START command executes one or more SQL or ISQL commands from the command buffer or a command file. The command(s) may contain parameters to which you assign values at execution time.

Scope

ISQL only.

ISQL Syntax

$$\text{STA}[\text{RT}]\,[\,CommandFileName\,]\,[\,(\,Value\,[\,,Value\,]\,[\,\dots\,]\,)\,]$$

Parameters

CommandFileName	identifies a command file. Name qualification follows HP-UX conventions:
	$[PathName/]\,FileName$
	Unless you specify an absolute path name, ISQL assumes that any path name you specify is relative to your current working directory. If *CommandFileName* is omitted, ISQL executes any command(s) currently in the command buffer.
Value	is a value to be substituted for a parameter in a command. Parameters in commands are identified with an ampersand (&) and a number from 1 through 100. Use a comma to separate values that correspond to different parameters. When you provide multiple values for a single parameter, separate those values with commas and enclose them in double quotation marks; for example, "value 1, value 2" are values for one parameter. The value provided for one parameter cannot exceed 80 bytes. If more values are specified than are needed, the unnecessary values are ignored.

Description

■ Within any single command file or command buffer, each parameter number can assume only one value or set of values. Therefore, you can refer to the same parameter more than once, but supply its value(s) only once. For example:

```
isql=> HOLD INFO &1\; SELECT &2 FROM &1;
isql=> START (PurchDB.Parts, "PartName,SalesPrice");
```

ISQL assigns the first value, *PurchDB.Parts*, to parameter &1 and the second value, *PartName,SalesPrice*, to parameter &2.

■ The following summarizes how parameter values you pass in the START command are interpreted by ISQL when a parameter is defined within single quotation marks (INFO '&1') or without them (INFO &1):

```
      HOW VALUE IS PASSED                    HOW VALUE IS INTERPRETED
                                              &1                '&1'
   START (   purchdb.parts   )          purchdb.parts       'purchdb.parts'
   START (  'purchdb.parts'  )          purchdb.parts       'purchdb.parts'
   START ("\'purchdb.parts\'")         'purchdb.parts'     ''purchdb.parts''
   START ('\'purchdb.parts\'')         'purchdb.parts'     ''purchdb.parts''
   START ('\"purchdb.parts\"')         "purchdb.parts"     '"purchdb.parts"'
```

Object names enclosed in double quotes cannot be split across lines.

■ To put commands in the command buffer, use one of the following ISQL commands:

HOLD
RECALL FILE
RECALL HISTORY

When commands are placed in the command buffer, ISQL concatenates the command lines.

■ When you use the START command to execute command files, two SET options affect processing:

□ If ECHO is on, commands and comments in the file are displayed as ISQL reads them. If ECHO is off, they are not displayed. Error messages and normal command output are always displayed.

□ If CONTINUE is on, ISQL continues processing commands from a file if an error is encountered. If CONTINUE is off, command file processing terminates if ISQL encounters an error. In either case, ISQL displays the command in error and an error message.

■ You can create command files with ISQL's EDIT or STORE commands. You can also create command files in the HP-UX environment using an editor. Put information to be processed in the first 80 bytes of each line only. Terminate each ISQL and SQL command in the file with a semicolon. Delimit comments with /* before the first comment character and */ after the last comment character. Commands and comments can span lines.

■ When executing command files containing parameters, ISQL prompts you for values if they are not supplied in the command line.

■ When you use the RECALL command to put a single command greater than 32K bytes into the command buffer, it is truncated and the START command cannot be used.

■ A command in the command buffer cannot be executed with a START in a command file.

■ You can include as many as nine recursive command files within a command file.

Authorization

You must have the corresponding authority for each SQL statement and ISQL command contained in the buffer or command file.

Example

```
isql=> EDIT MyFile;

    ⋮


/*This file updates statistics of table &1
and does several SELECT operations.*/

UPDATE STATSTICS FOR TABLE &1;
SELECT &2 FROM &1;
PRINT;
END;
SELECT &2 FROM &1 WHERE &3;
PRINT;
END;
COMMIT WORK;
:wq MyFile
isql=> SET ECHO ON; SET CONTINUE ON;
```

These SET options affect command file processing as follows:
All commands and error messages are displayed. Processing
continues if an error is encountered.

```
isql=> START MyFile (PurchDB.Inventory, "PartNumber,QtyOnHand",
> "QtyOnHand <100");

/*This file updates statistics of table &1
and does several SELECT operations.*/

UPDATE STATSTICS FOR TABLE &1;

UPDATE STATSTICS FOR TABLE PurchDB.Inventory;
                  |
Unexpected keyword.  (DBERR 1006)
SELECT &2 FROM &1;
```

First group of selected rows of query result are displayed.

```
PRINT;
```

Number of rows selected is 20.

```
END;
SELECT &2 FROM &1 WHERE &3;
```

First group of data rows of query result are displayed.

```
PRINT;
```

Number of rows selected is 20.

```
END;
COMMIT WORK;
isql=> SET ECHO OFF; SET CONTINUE OFF;
```

These SET options affect command file processing as follows:
Only error messages are displayed. Processing terminates if an
error is encountered.

```
isql=> START MyFile (PurchDB.Parts,*,SalesPrice> 1000);

UPDATE STATSTICS FOR TABLE PurchDB.Parts;
                  |
Unexpected keyword.  (DBERR 1006)
Command file execution terminated.  (DBERR 60)
Last command:
UPDATE STATSTICS FOR TABLE &1;
isql=>
```

STORE

The STORE command saves the contents of the command buffer in a command file.

Scope

ISQL only.

ISQL Syntax

STO[RE]*FileName* [R[EPLACE]]

Parameters

FileName   identifies the file that will contain the contents of the command
                        buffer. File name qualification follows HP-UX conventions:

> [*PathName*/]*FileName*

Unless you specify an absolute path name, ISQL assumes that
any path name you specify is relative to your current working
directory.

REPLACE   overwrites any existing file with the same name as *FileName*.

Description

■ You can also create command files from within ISQL with the EDIT command or from
HP-UX using an editor. ISQL processes only up to the first 80 bytes of each line in the
command file.

■ You can put commands in the command buffer with these ISQL commands:

> HOLD
> RECALL FILE
> RECALL HISTORY

■ In prompting mode, ISQL prompts you for a file name as follows:

```
isql=> STORE;
File name> FileName;
isql=>
```

If the file already exists, ISQL prompts you as to whether to replace the existing file.

Example

```
isql=> RECALL FILE Monthly;

UPDATE STATISTICS FOR TABLE PurchDB.OrderItems; ...; UP
DATE STATISTICS FOR TABLE PurchDB.Inventory;

isql=> CHANGE /Inventory/Orders/;

UPDATE STATISTICS FOR TABLE PurchDB.OrderItems; ...; UP
DATE STATISTICS FOR TABLE PurchDB.Orders;

isql=> STORE Monthly REPLACE;
isql=>
```

SYSTEM

The SYSTEM command lets you execute HP-UX commands from within ISQL.

Scope

ISQL only.

ISQL Syntax

$$\left\{ \begin{array}{c} \mathrm{SY[STEM]} \\ ! \end{array} \right\} \; [\; \mathit{HP\text{-}UXcommand} \;]$$

Parameters

HP-UXcommand   is any HP-UX command. If entered, it is passed to HP-UX to be executed, and you return to ISQL automatically. If omitted, the HP-UX prompt appears, and you can execute HP-UX commands until you enter EXIT to return to ISQL.

Example

```
isql=> !more Daily;
UPDATE STATISTICS FOR TABLE PurchDB.OrderItems;
UPDATE STATISTICS FOR TABLE PurchDB.Orders;
isql=> !
$ date
Wed Aug 26  12:52:03 PDT 1987
$ pwd
/users/dbsupport/thomas/sql
$ exit
isql=>
```

UNLOAD

The UNLOAD command copies data from one or more tables into an output file.

Scope

ISQL only.

ISQL Syntax

$$U[NLOAD]\ TO\ \left\{ \begin{array}{l} E[XTERNAL] \\ I[NTERNAL] \end{array} \right\}\ OutputFileName$$

$$FROM\ \left\{ \begin{array}{l} [Owner.]TableName \\ [Owner.]ViewName \\ "SelectStatement" \end{array} \right\}\ ExternalOutputSpec$$

Parameters

EXTERNAL	indicates that the output file is a text file; files in this format are intended for use by user programs. This type of unloading allows data to be easily manipulated.
INTERNAL	indicates that the output file is in a format recognizable only by ALLBASE/SQL; files in this format can later be loaded with the LOAD command's INTERNAL option. This format allows data to be loaded faster by ALLBASE/SQL. However, this type of unloading should *not* be used for migration.
OutputFileName	identifies the output file. The file name must follow HP-UX naming conventions:

> [*PathName*/]*FileName*

Unless you specify an absolute path name, ISQL assumes any path name you specify is relative to your current working directory.

If you specify the name of an existing file, the existing file is overwritten.

[*Owner.*]*TableName*	identifies the table from which you want to unload data. If you specify this option, all columns and rows from the table are unloaded, in the order specified in the table definition.
[*Owner.*]*ViewName*	identifies a view to unload from.
SelectStatement	is an ALLBASE/SQL SELECT statement that identifies one or more tables in the DBEnvironment you are using from which you want to unload data. The *SelectStatement* may specify criteria for limiting columns and rows to be unloaded. Columns are unloaded in the order specified in the select list. The SELECT statement must be enclosed in double quotation marks.

ExternalOutputSpec   describes an EXTERNAL file. The syntax for this option is:

> *DescriptionFileName* {*OutputLength* [*FractionLength*] [*NullIndicator*]} [...]

Description-   identifies a description file. The file name follows
FileName   the same conventions as the name of the output
        file.

The description file contains at least one line for each column unloaded. The first line describes the first column unloaded, the second line describes the second column unloaded, and so on. Each line contains seven fields. Data in all the fields except the column name field is right-justified; the column name is left-justified.

Column(s)	Contents
1-20	Column name.

25	LOAD/UNLOAD type code (see following list):
	
	  0 = SMALLINT & INTEGER
	  1 = BINARY
	  2 = CHAR
	    DATE[1]
	    TIME[1]
	    DATETIME[1]
	    INTERVAL[1]
	  3 = VARCHAR
	  4 = FLOAT
	  5 = DECIMAL
	  6 = TID (tuple identifier)
	  8 = NATIVE CHAR[2]
	  9 = NATIVE VARCHAR[2]
	 14 = VARBINARY
	 15 = LONG BINARY
	 16 = LONG VARBINARY
	
	[1] DATE, TIME, DATETIME and INTERVAL columns are known to the LOAD and UNLOAD commands as CHAR data.
	
	[2] NATIVE CHAR or VARCHAR is what SQLCore uses internally when a CHAR or VARCHAR column is defined with a LANG= clause.

30-40	Output length.
41-50	Fraction length (number of decimal places).
51-60	Starting location of data.
61-70	Starting location of length of VARCHAR data.
75	Null indicator.

OutputLength   is the number of columns (bytes) to allocate in the output line for data from each column unloaded. Specify *OutputLength* (and, optionally, *FractionLength* and *NullIndicator*) for each column to be unloaded in the order in which they are to be unloaded. For numeric values, allow one additional space for the sign. For float and decimal numbers, allow one extra space for the decimal point.

If the output length is smaller than the actual column length, CHAR and VARCHAR data is truncated and question marks are written instead of numeric data. Fixed-length fields are written to their maximum length; unused space is filled with blanks.

The actual length of VARCHAR data is prefixed to the data as a 10-byte field.

Fraction-   is the number of decimal places to allocate. You
Length   must specify a *FractionLength* for data of type FLOAT or DECIMAL. Conversely, you may not specify this attribute for data of other types.

NullIndicator   is the character to insert in the external file when ISQL encounters a null value. You must specify a *NullIndicator* for any column that can contain null values. The null indicator can be any one-byte character except a blank, a semicolon, a single or double quotation mark, a minus sign, or the current SET command's ESCAPE character. If you use a slash (/), precede it with the current ISQL escape character. ISQL uses the null indicator to mark a null value in the output file when a given column is null (or empty).

UNLOAD

Description

■ You must establish a DBE session with a CONNECT or a START DBE command before using the UNLOAD command.

■ You can unload an external file directly to tape. For example:

```
isql=>UNLOAD;
Output file format (i[nternal] or e[xternal])> external;
Output file name> /dev/rmt/0m;
TableName or "SelectStatement"> ManufDB.TestData;
Command in progress.
Number of rows processed is 12
```

The description file needs to be a separate file, and cannot be unloaded directly to tape.

■ If you are unloading to tape and using multiple tapes, you are notified when you reach the end of each tape. At this point, dismount the current tape from the tape drive and mount the next tape. When the new tape is online, enter a Y at the ISQL prompt to continue the UNLOAD process. Alternatively, you can stop at this point by entering N at the prompt. Number the tapes if record order is important. When reloading them, each tape has to be loaded separately.

■ When unloading has begun, ISQL displays the following message:

```
Command in progress.
```

■ During the unloading, ISQL displays as follows the cumulative number of rows unloaded as it unloads each group of rows; this number does not necessarily equal the number of rows unloaded when you are unloading to an internal file:

```
Number of rows processed is n
```

With UNLOAD EXTERNAL, the number of rows equals the number of records. With UNLOAD INTERNAL, the number of records is greater than the number of rows unloaded.

■ In prompting mode, ISQL prompts you for the output options:

```
isql=> UNLOAD;

Output file format (i[nternal] or e[xternal])>  Enter INTERNAL; or EXTERNAL;.
Output file name> OutputFileName;
TableName or "SelectStatement"> Enter Tablename; or "SelectStatement";
```

■ UNLOAD INTERNAL may not be used with LONG columns.

■ When you are unloading to an external file in prompting mode, ISQL also prompts you for a description file name and for information on each column to be unloaded.

```
Description file name> FileName;
Column COLUMNNAME1, Type (Size):
Output length> OutputLength;
Fraction length> FractionLength;
Null indicator> NullIndicator;
Column COLUMNNAME2, Type (Size):

    ⋮
```

You are prompted for a fraction length only if the column contains DECIMAL or FLOAT data. You are prompted for a null indicator only if the column allows null values.

■ When you are prompted for the length of a LONG column in using the UNLOAD EXTERNAL command, respond with the length of the output device, not the size of the LONG data type. The length of the output device is shown as the size of the LONG column in the description that precedes the prompt. Example:

```
Column c2, Long Binary (46):
Output length> 46;
```

Note that the value 46 is the maximum length of the output device plus the option or options prefixed to it. You may choose a different size if you wish.

■ Within a SELECTSTATEMENT, if you wish to refer to an object name that was created with double quotes, you must precede each inner quote with an escape character, as in the following example:

```
UNLOAD TO EXTERNAL EParts FROM
"Select * FROM \"PurchDB\".Parts";
```

To prevent any possible conflict of double-quoted strings, avoid the use of double-quoted object names.

■ Object names enclosed in double quotes cannot be split across lines.

■ External files containing LONG columns display only the output device name, not the data. The output device itself will contain the actual data which is selected during the UNLOAD.

■ In external files, DATE, TIME, DATETIME, and INTERVAL columns appear as characters. Internally, they are stored as binary values, although ISQL returns a code of 2 (CHAR) for them in creating description files.

Authorization

You must have the authority to select from the table(s) named. Refer to the SELECT statement in the *ALLBASE/SQL Reference Manual*.

UNLOAD

Example

```
isql=> UNLOAD;

Output file format (i[nternal] or e[xternal])> e>
Output file name> EXTD;
TableName or "SelectStatement"> ManufDB.TestData;
Description file name> DTD;
Column BATCHSTAMP, DateTime (23):
Output length> 24;
Column TESTDATE, Date (10):
Output length> 11;
Null indicator> ?
Column TESTSTART, Time (8):
Output length> 9
Null indicator> ?
Column TESTEND, Time (8):
Output length> 9
Null indicator> ?
Column LABTIME, Interval (20):
Output length> 20;
Null indicator> ?
Column PASSQTY, Integer:
Output length> 3
Null indicator> ?
Column TESTQTY, Integer:
Output length> 3
Null indicator> ?
Command in progress.
Number of rows processed is 12

isql=>
```

```
isql=> LIST FILE EXTD;

--------------
|   EXTD     |
--------------
1984-06-19 08:45:33.123 1984-06-23 08:12:19 13:23:01        0 05:10:42.000 49 50
1984-06-14 11:13:15.437 1984-06-17 08:05:02 14:01:27        0 05:56:25.000 47 50
1984-07-02 14:54:07.984 1984-07-05 14:03:21 19:33:54        0 05:30:33.000 48 50
1984-07-22 09:06:23.319 1984-07-29 14:01:28 20:16:07        0 06:14:39.000 50 50
1984-06-19 08:45:33.123 1984-06-27 08:02:29 14:13:31        0 06:11:02.000 49 50
1984-07-09 16:07:17.394 1984-07-13 08:43:16 13:22:44        0 04:39:28.000 46 50
1984-07-13 09:25:53.183 1984-07-18 14:07:01 20:03:22        0 05:56:21.000 49 50
1984-07-15 13:22:13.782 1984-07-22 09:01:48 14:47:02        0 05:45:14.000 50 50
1984-07-09 16:07:17.394 1984-07-19 08:13:26 13:45:34        0 05:32:08.000 49 50
1984-07-15 15:32:03.529 1984-07-23 14:02:34 19:56:02        0 05:53:28.000 49 50
1984-07-25 10:15:58.159 1984-07-30 08:25:11 13:34:22        0 05:09:11.000 48 50
1984-07-25 10:15:58.159 1984-08-02 08:01:13 14:29:03        0 06:27:50.000 47 50
1984-08-19 08:45:33.123 1984-08-25 08:12:19 19:30:00        5 04:23:00.090 49 50

isql=> LIST FILE DTD;

--------------
|   DTD      |
--------------
BATCHSTAMP              2              24          0          1          0
TESTDATE                2              11          0          25         0    ?
TESTSTART               2              9           0          36         0    ?
TESTEND                 2              9           0          45         0    ?
LABTIME                 2              20          0          54         0    ?
PASSQTY                 0              3           0          75         0    ?
TESTQTY                 0              3           0          78         0    ?
ENDATA

isql=> unload;

Output file format (i[nternal] or e[xternal])> internal;
Output file name> DTD;
TableName or "SelectStatement"> ManufDB.TestData;
Command in progress.
Number of rows processed is 13

isql=>
```

UNLOAD

```
isql=> UNLOAD TO EXTERNAL EXTD FROM ManufDB.TestData
> DTD
> 24
> 11 ?
> 9  ?
> 9  ?
> 20 ?
> 3  ?
> 3  ?;

Command in progress.
Number of rows processed is 13

isql=>
```

The same command in a script file would look like this:

```
UNLOAD TO EXTERNAL EXTD FROM ManufDB.TestData
DTD 24 11 ? 9 ? 9 ? 20 ? 3 ? 3 ?;
```

SQL Syntax Summary

ADD DBEFILE

ADD DBEFILE $DBEFileName$ TO DBEFILESET $DBEFileSetName$

ADD TO GROUP

$$\text{ADD} \left\{ \begin{array}{l} DBEUserID \\ GroupName \\ ClassName \end{array} \right\} [\ ,\ \ldots\ ] \text{TO GROUP } TargetGroupName$$

ADVANCE

$$\text{ADVANCE } CursorName \left[\text{USING } [\text{SQL}]\text{DESCRIPTOR} \left\{ \begin{array}{l} \text{SQLDA} \\ AreaName \end{array} \right\} \right]$$

ALTER DBEFILE

$$\text{ALTER DBEFILE } DBEFileName \text{ SET TYPE} = \left\{ \begin{array}{l} \text{TABLE} \\ \text{INDEX} \\ \text{MIXED} \end{array} \right\}$$

ALTER TABLE

$$\text{ALTER TABLE } [\ Owner.\ ] TableName \left\{ \begin{array}{l} AddColumnSpecification \\ AddConstraintSpecification \\ DropConstraintSpecification \\ SetTypeSpecification \\ SetPartitionSpecification \end{array} \right\}$$

AddColumnSpecification

$$\text{ADD} \left\{ \begin{array}{l} (\ ColumnDefinition\ [\ ,\ \ldots\ ]\ ) \\ Column\ Definition \end{array} \right\} [\ \text{CLUSTERING ON CONSTRAINT}\ [\ ConstraintID\ ]\ ]$$

AddConstraintSpecification

$$\text{ADD CONSTRAINT} \left\{ \left(\left\{ \begin{array}{l} UniqueConstraint \; [\text{CONSTRAINT } ConstraintID1] \\ ReferentialConstraint \; [\text{CONSTRAINT } ConstraintID1] \\ CheckConstraint \; [\text{CONSTRAINT } ConstraintID1] \end{array} \right\} [\, , \; \ldots \;] \right) \right\}$$
$$[\text{CLUSTERING ON CONSTRAINT } [\, ConstraintID2 \,]]$$

DropConstraintSpecification

$$\text{DROP CONSTRAINT} \left\{ \begin{array}{l} (ConstraintID \; [\, , \; \ldots \;]) \\ ConstraintID \end{array} \right\}$$

SetTypeSpecification

$$\text{SET TYPE} \left\{ \begin{array}{l} \text{PRIVATE} \\ \text{PUBLICREAD} \\ \text{PUBLIC} \\ \text{PUBLICROW} \end{array} \right\} \left[\begin{array}{l} \text{RESET AUTHORITY} \\ \text{PRESERVE AUTHORITY} \end{array} \right]$$

SetPartitionSpecification

$$\text{SET PARTITION} \left\{ \begin{array}{l} PartitionName \\ \text{DEFAULT} \\ \text{NONE} \end{array} \right\}$$

Assignment (=)

$$\left\{ \begin{array}{l} : LocalVariable \\ : ProcedureParameter \end{array} \right\} = Expression;$$

BEGIN

$$\text{BEGIN} \; [\, statement; \,] [\; \ldots \;] \text{END};$$

BEGIN ARCHIVE

$$\text{BEGIN ARCHIVE}$$

BEGIN DECLARE SECTION

$$\text{BEGIN DECLARE SECTION}$$

BEGIN WORK

$$\text{BEGIN WORK} \; [\, Priority \,] \left[\begin{array}{l} \text{RR} \\ \text{CS} \\ \text{RC} \\ \text{RU} \end{array} \right] \left[\text{LABEL} \left\{ \begin{array}{l} ' LabelString' \\ : HostVariable \end{array} \right\} \right] \left[\left[\begin{array}{l} \text{PARALLEL} \\ \text{NO} \end{array} \right] \text{FILL} \right]$$

CHECKPOINT

$$\texttt{CHECKPOINT} \begin{bmatrix} :HostVariable \\ :LocalVariable \\ :ProcedureParameter \end{bmatrix}$$

CLOSE

$$\texttt{CLOSE}\ CursorName \left[\texttt{USING} \left\{ \begin{array}{l} [\texttt{SQL}]\texttt{DESCRIPTOR} \left\{ \begin{array}{l} \texttt{SQLDA} \\ AreaName \end{array} \right\} \\ :HostVariable\ [\,[\texttt{INDICATOR}]:Indicator\,]\,[\,,\ \ldots\ ] \end{array} \right\} \right]$$

COMMIT ARCHIVE

```
COMMIT ARCHIVE
```

COMMIT WORK

$$\texttt{COMMIT WORK}\ [\texttt{RELEASE}]$$

CONNECT

$$\texttt{CONNECT TO} \left\{ \begin{array}{l} 'DBEnvironmentName' \\ :HostVariable1 \end{array} \right\} \left[\texttt{AS} \left\{ \begin{array}{l} 'ConnectionName' \\ :HostVariable2 \end{array} \right\} \right]$$

$$\left[\texttt{USER} \left\{ \begin{array}{l} 'UserID' \\ :HostVariable3 \end{array} \right\} [\texttt{USING}\ :HostVariable4] \right]$$

CREATE DBEFILE

$$\texttt{CREATE DBEFILE}\ DBEFilename\ \texttt{WITH PAGES} = DBEFileSize,\ \texttt{NAME} = 'SystemFileName'$$
$$[\,,\ \texttt{INCREMENT} = DBEFileIncrSize[\,,\ \texttt{MAXPAGES} = DBEFileMaxSize\,]\,]$$

$$\left[,\texttt{TYPE} = \left\{ \begin{array}{l} \texttt{TABLE} \\ \texttt{INDEX} \\ \texttt{MIXED} \end{array} \right\} \right]$$

CREATE DBEFILESET

$$\texttt{CREATE DBEFILESET}\ DBEFileSetName$$

CREATE GROUP

$$\texttt{CREATE GROUP}\ [\,Owner.\,]\,GroupName$$

CREATE INDEX

CREATE [UNIQUE] [CLUSTERING] INDEX [$Owner.$] $IndexName$ ON
[$Owner.$] $TableName$ ($\left\{ ColumnName \begin{bmatrix} \texttt{ASC} \\ \texttt{DESC} \end{bmatrix} \right\}$ [, ...])

CREATE PARTITION

CREATE PARTITION $PartitionName$ WITH ID = $PartitionNumber$

CREATE PROCEDURE

CREATE PROCEDURE [$Owner.$] $ProcedureName$ [LANG = $ProcLangName$]
[($ParameterDeclaration$ [, $ParameterDeclaration$] [...])]
[WITH RESULT $ResultDeclaration$ [, $ResultDeclaration$] [...]]
AS BEGIN [$ProcedureStatement$] [...] END [IN $DBEFileSetName$]

ParameterDeclaration

$ParameterName$ $ParameterType$ [LANG = $ParameterLanguage$]
[DEFAULT $DefaultValue$] [NOT NULL] [OUTPUT [ONLY]]

ResultDeclaration

$ResultType$ [LANG = $ResultLanguage$] [NOT NULL]

CREATE RULE

CREATE RULE [$Owner.$] $RuleName$

AFTER $StatementType$ [, ...] $\left\{ \begin{matrix} \texttt{ON} \\ \texttt{OF} \\ \texttt{FROM} \\ \texttt{INTO} \end{matrix} \right\}$ [$Owner$] . $TableName$

$\left[\texttt{REFERENCING} \left\{ \begin{matrix} \texttt{OLD AS } OldCorrelationName \\ \texttt{NEW AS } NewCorrelationName \end{matrix} \right\} [...] \right]$ [WHERE $FiringCondition$]
EXECUTE PROCEDURE [$OwnerName.$] $ProcedureName$ [($ParameterValue$ [, ...])]
[IN $DBEFileSetName$]

CREATE SCHEMA

CREATE SCHEMA AUTHORIZATION $AuthorizationName$ $\begin{bmatrix} TableDefinition \\ ViewDefinition \\ IndexDefinition \\ ProcedureDefinition \\ RuleDefinition \\ CreateGroup \\ AddToGroup \\ GrantStatement \end{bmatrix}$ [...]

CREATE TABLE

$$
\texttt{CREATE} \begin{bmatrix} \texttt{PRIVATE} \\ \texttt{PUBLICREAD} \\ \texttt{PUBLIC} \\ \texttt{PUBLICROW} \end{bmatrix} \texttt{TABLE} \; [\,Owner.\,]\; TableName
$$

$$
[\,\texttt{LANG =}\; TableLanguageName\,]
$$

$$
(\; \left\{ \begin{array}{l} ColumnDefinition \\ UniqueConstraint \\ ReferentialConstraint \\ CheckConstraint \end{array} \right\} [\,,\; \ldots\;] \;)
$$

$$
\begin{bmatrix} \texttt{UNIQUE HASH ON}\; (HashColumnName\; [\,,\; \ldots\;]) \;\texttt{PAGES =}\; PrimaryPages \\ \texttt{HASH ON CONSTRAINT}\; [\,ConstraintID\,]\texttt{PAGES =}\; PrimaryPages \\ \texttt{CLUSTERING ON CONSTRAINT}\; [\,ConstraintID\,] \end{bmatrix}
$$

$$
\begin{bmatrix} \texttt{IN PARTITION} \left\{ \begin{array}{l} PartitionName \\ \texttt{DEFAULT} \\ \texttt{NONE} \end{array} \right\} \end{bmatrix}
$$

$$
[\,\texttt{IN}\; DBEFileSetName1\,]
$$

Column Definition

$$
ColumnName \left\{ \begin{array}{l} ColumnDataType \\ LongColumnType\; [\,\texttt{IN}\; DBEFileSetName2\,] \end{array} \right\}
$$

$$
[\,\texttt{LANG =}\; ColumnLanguageName\,]
$$

$$
[\,[\,\texttt{NOT}\,]\texttt{CASE SENSITIVE}\,]
$$

$$
\begin{bmatrix} \texttt{DEFAULT} \left\{ \begin{array}{l} Constant \\ \texttt{USER} \\ \texttt{NULL} \\ CurrentFunction \end{array} \right\} \end{bmatrix}
$$

$$
\begin{bmatrix} \texttt{NOT NULL} \; \begin{bmatrix} \left\{ \begin{array}{l} \texttt{UNIQUE} \\ \texttt{PRIMARY KEY} \end{array} \right\} [\,\texttt{CONSTRAINT}\; ConstraintID\,] \end{bmatrix} \\ \texttt{REFERENCES}\; RefTableName\; [\,(RefColumnName)\,]\,[\,\texttt{CONSTRAINT}\; ConstraintID\,] \\ [\;\ldots\;] \\ \\ \texttt{CHECK}\; (SearchCondition)\; [\,\texttt{CONSTRAINT}\; ConstraintID\,] \\ [\,\texttt{IN}\; DBEFileSetName3\,] \end{bmatrix}
$$

$$
[\;\ldots\;]
$$

Unique Constraint (Table Level)

$$
\left\{ \begin{array}{l} \texttt{UNIQUE} \\ \texttt{PRIMARY KEY} \end{array} \right\} (\; ColumnName\; [\,,\; \ldots\;]) \; [\,\texttt{CONSTRAINT}\; ConstraintID\,]
$$

Referential Constraint (Table Level)

$$
\texttt{FOREIGN KEY}\; (\; FKColumnName\; [\,,\; \ldots\;])
$$

$$
\texttt{REFERENCES}\; RefTableName\; [\,(\; RefColumnName\; [\,,\; \ldots\;])\,]\; [\,\texttt{CONSTRAINT}\; ConstraintID\,]
$$

Check Constraint (Table Level)

CHECK ($SearchCondition$) [CONSTRAINT $ConstraintID$] [IN $DBEFileSetName3$]

CREATE TEMPSPACE

CREATE TEMPSPACE $TempSpaceName$
WITH [MAXFILEPAGES = $MaxTempFileSize$,] LOCATION =' $PhysicalLocation$'

CREATE VIEW

CREATE VIEW [$Owner.$] $ViewName$ [($ColumnName$ [, ...])]
AS $QueryExpression$ [IN $DBEFileSetName$]
[WITH CHECK OPTION [CONSTRAINT $ConstraintID$]]

DECLARE CURSOR

DECLARE $CursorName$ [IN $DBEFileSetName$] CURSOR FOR

$$
\left\{ \begin{array}{l} \left\{ \begin{array}{l} QueryExpression \\ SelectStatementName \end{array} \right\} \left[\begin{array}{l} \texttt{FOR UPDATE OF } \{ ColumnName \} [, ...] \\ \texttt{FOR READ ONLY} \end{array} \right] \\ ExecuteProcedureStatement \\ ExecuteStatementName \end{array} \right\}
$$

DECLARE $Variable$

DECLARE $\{ LocalVariable \}$ [, ...] $VariableType$ [LANG = $VariableLangName$]

$$
\left[\texttt{DEFAULT} \left\{ \begin{array}{l} Constant \\ \texttt{NULL} \\ CurrentFunction \end{array} \right\} \right] [\texttt{NOT NULL}]
$$

DELETE

DELETE FROM $\left\{ \begin{array}{l} [Owner.] TableName \\ [Owner.] ViewName \end{array} \right\}$ [WHERE $SearchCondition$]

DELETE WHERE CURRENT

DELETE FROM $\left\{ \begin{array}{l} [Owner.] TableName \\ [Owner.] ViewName \end{array} \right\}$ WHERE CURRENT OF $CursorName$

DESCRIBE

DESCRIBE $\left[\begin{array}{l} \texttt{OUTPUT} \\ \texttt{INPUT} \\ \texttt{RESULT} \end{array} \right]$ $StatementName$ $\left\{ \begin{array}{l} \texttt{INTO} [[\texttt{SQL}]\texttt{DESCRIPTOR}] \\ \texttt{USING} [\texttt{SQL}]\texttt{DESCRIPTOR} \end{array} \right\}$ $\left\{ \begin{array}{l} \texttt{SQLDA} \\ AreaName \end{array} \right\}$

DISABLE AUDIT LOGGING

```
DISABLE AUDIT LOGGING
```

DISABLE RULES

```
DISABLE RULES
```

DISCONNECT

$$\text{DISCONNECT} \left\{ \begin{array}{l} \text{'}ConnectionName\text{'} \\ \text{'}DBEnvironmentName\text{'} \\ :HostVariable \\ \text{ALL} \\ \text{CURRENT} \end{array} \right\}$$

DROP DBEFILE

$$\text{DROP DBEFILE } DBEFileName$$

DROP DBEFILESET

$$\text{DROP DBEFILESET } DBEFileSetName$$

DROP GROUP

$$\text{DROP GROUP } GroupName$$

DROP INDEX

$$\text{DROP INDEX } [\,Owner.\,]IndexName\,[\text{FROM } [\,Owner.\,]TableName\,]$$

DROP MODULE

$$\text{DROP MODULE } [\,Owner.\,]ModuleName\,[\text{PRESERVE}\,]$$

DROP PARTITION

$$\text{DROP PARTITION } PartitionName$$

DROP PROCEDURE

$$\text{DROP PROCEDURE } [\,Owner.\,]ProcedureName\,[\text{PRESERVE}\,]$$

DROP RULE

DROP RULE [*Owner.*] *RuleName* [FROM TABLE [*Owner.*] *TableName*]

DROP TABLE

DROP TABLE [*Owner.*] *TableName*

DROP TEMPSPACE

DROP TEMPSPACE *TempSpaceName*

DROP VIEW

DROP VIEW [*Owner.*] *ViewName*

ENABLE AUDIT LOGGING

ENABLE AUDIT LOGGING

ENABLE RULES

ENABLE RULES

END DECLARE SECTION

END DECLARE SECTION

EXECUTE

$$
\text{EXECUTE} \left\{ \begin{array}{l} StatementName \\ [\,Owner.\,]\,ModuleName\,[\,(SectionNumber)\,] \end{array} \right\}
$$

$$
\left[\text{USING} \left\{ \begin{array}{l} [\,\text{SQL}\,]\text{DESCRIPTOR} \left\{ \begin{array}{l} [\,\text{INPUT}\,] \left\{ \begin{array}{l} \text{SQLDA} \\ AreaName1 \end{array} \right\} \left[\text{AND OUTPUT} \left\{ \begin{array}{l} \text{SQLDA} \\ AreaName2 \end{array} \right\} \right] \\ \text{OUTPUT} \left\{ \begin{array}{l} \text{SQLDA} \\ AreaName \end{array} \right\} \end{array} \right\} \\ [\,\text{INPUT}\,]\,HostVariableSpecification1\,[\,\text{AND OUTPUT}\,HostVariableSpecification2\,] \\ \text{OUTPUT}\,HostVariableSpecification \\ :Buffer\,[\,,:StartIndex\,[\,,\,:NumberOfRows\,]\,] \end{array} \right\} \right]
$$

HostVariableSpecification

:*HostVariableName*[[INDICATOR] :*IndicatorVariable*][, ...]

EXECUTE IMMEDIATE

$$\texttt{EXECUTE IMMEDIATE} \left\{ \begin{array}{l} \textit{'String'} \\ : \textit{HostVariable} \end{array} \right\}$$

EXECUTE PROCEDURE

$$\texttt{EXECUTE PROCEDURE} \; [\, : \textit{ReturnStatusVariable} \; \texttt{=} \;] \; [\, \textit{Owner.}\,] \textit{ProcedureName}$$
$$[\, (\, [\, \textit{ActualParameter}\,]\, [\, \texttt{,} \; [\, \textit{ActualParameter}\,]\,]\, [\, \dots\,]\,)\,]$$

ActualParameter

$$[\, \textit{ParameterName} \; \texttt{=} \;] \textit{ParameterValue} \; [\, \texttt{OUTPUT} \; [\, \texttt{ONLY}\,]\,]$$

FETCH

$$[\, \texttt{BULK}\,]\texttt{FETCH} \; \textit{CursorName} \left\{ \begin{array}{l} \texttt{INTO} \; \textit{HostVariableSpecification} \\ \texttt{USING} \left\{ \begin{array}{l} [\, \texttt{SQL}\,]\texttt{DESCRIPTOR} \left\{ \begin{array}{l} \texttt{SQLDA} \\ \textit{AreaName} \end{array} \right\} \\ \textit{HostVariableSpecification} \end{array} \right\} \end{array} \right\}$$

BULK HostVariableSpecification

$$: \textit{Buffer} \; [\, , : \textit{StartIndex} \; [\, , : \textit{NumberOfRows}\,]\,]$$

Non-BULK HostVariableSpecification

$$\left\{ \; : \textit{HostVariable} \; [\, [\, \texttt{INDICATOR}\,] : \textit{Indicator}\,]\, \right\} [\, , \; \dots\;]$$

GENPLAN

$$\texttt{GENPLAN} \; [\, \texttt{WITH} \; (\, \textit{VariableName} \; \textit{DataType} \; [\, , .. \,]\,)\,]$$
$$\texttt{FOR} \; \textit{SQLStatement}$$

GOTO

$$\left\{ \begin{array}{l} \texttt{GOTO} \\ \texttt{GO TO} \end{array} \right\} \left\{ \begin{array}{l} \textit{Label} \\ \textit{Integer} \end{array} \right\}$$

GRANT

$$\text{GRANT} \left\{ \begin{array}{l} \text{ALL } [\text{PRIVILEGES}] \\ \left\{ \begin{array}{l} \text{SELECT} \\ \text{INSERT} \\ \text{DELETE} \\ \text{ALTER} \\ \text{INDEX} \\ \text{UPDATE } [(\{ColumnName\}[,\ \ldots\ ])] \\ \text{REFERENCES } [(\{ColumnName\}[,\ \ldots\ ])] \end{array} \right\} \Big|,\ \ldots\ \Big| \end{array} \right\}$$

$$\text{ON} \left\{ \begin{array}{l} [\ Owner.\ ]\ TableName \\ [\ Owner.\ ]\ ViewName \end{array} \right\} \text{TO} \left\{ \begin{array}{l} DBEUserID \\ GroupName \\ ClassName \\ \text{PUBLIC} \end{array} \right\} [,\ \ldots\ ] \ [\text{WITH GRANT OPTION}]$$

$$\left[\text{BY} \left\{ \begin{array}{l} DBEUserID \\ ClassName \end{array} \right\} \right]$$

Grant RUN or EXECUTE Authority

$$\text{GRANT} \left\{ \begin{array}{l} \text{RUN ON } [\ Owner.\ ] ModuleName \\ \text{EXECUTE ON PROCEDURE } [\ Owner.\ ] ProcedureName \end{array} \right\} \text{TO}$$

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} DBEUserID \\ GroupName \\ ClassName \end{array} \right\} [,\ \ldots\ ] \\ \text{PUBLIC} \end{array} \right\}$$

Grant CONNECT, DBA, or RESOURCE Authority

$$\text{GRANT} \left\{ \begin{array}{l} \text{CONNECT} \\ \text{DBA} \\ \text{RESOURCE} \end{array} \right\} \text{TO} \left\{ \begin{array}{l} DBEUserID \\ GroupName \\ ClassName \end{array} \right\} [,\ \ldots\ ]$$

Grant DBEFileSet Authority

$$\text{GRANT} \left\{ \begin{array}{l} \text{SECTIONSPACE} \\ \text{TABLESPACE} \end{array} \right\} [,\ \ldots\ ] \ \text{ON DBEFILESET } DBEFileSetName \text{ TO}$$

$$\left\{ \begin{array}{l} DBEUserID \\ GroupName \\ ClassName \\ \text{PUBLIC} \end{array} \right\} [,\ \ldots\ ]$$

IF

IF $Condition$ THEN $[\ Statement;\ [\ \ldots\ ]\ ]$
$[\ \text{ELSEIF}\ Condition\ \text{THEN}\ [\ Statement;\ [\ \ldots\ ]\ ]\ ]$
$[\ \text{ELSE}\ [\ Statement;\ [\ \ldots\ ]\ ]\ ]$ ENDIF;

INCLUDE

$$\text{INCLUDE} \left\{ \begin{array}{l} \text{SQLCA} \; [\,[\,\text{IS}\,]\,\text{EXTERNAL}\,] \\ \text{SQLDA} \end{array} \right\}$$

INSERT - 1

$$[\,\text{BULK}\,]\,\text{INSERT INTO} \left\{ \begin{array}{l} [\,Owner.\,]\,TableName \\ [\,Owner.\,]\,ViewName \end{array} \right\}$$

$$[\,(\,\{\,ColumnName\,\}\,[\,,\,\ldots\,]\,)\,]$$

$$\text{VALUES} \; (\left\{ \begin{array}{l} SingleRowValues \\ BulkValues \\ \text{?} \end{array} \right\})$$

SingleRowValues

$$\left\{ \begin{array}{l} \text{NULL} \\ \text{USER} \\ : HostVariable \; [\,[\,\text{INDICATOR}\,]\,:IndicatorVariable\,] \\ \text{?} \\ : LocalVariable \\ : ProcedureParameter \\ :: Built\text{-}inVariable \\ ConversionFunction \\ CurrentFunction \\ \left[\begin{array}{c} + \\ - \end{array} \right] \left\{ \begin{array}{l} Integer \\ Float \\ Decimal \end{array} \right\} \\ {'}CharacterString{'} \\ \text{0x}\,HexadecimalString \\ {'}LongColumnIOString{'} \end{array} \right\} \; [\,,\,\ldots\,]$$

LongColumnIOString

$$< \left\{ \begin{array}{l} [\,PathName\text{/}\,]\,FileName \\ \text{%}\; SharedMemoryAddress \end{array} \right\}$$

$$\left[\begin{array}{l} \left\{ \begin{array}{l} > \\ >> \\ >! \end{array} \right\} [\,PathName\text{/}\,] \left\{ \begin{array}{l} FileName \\ CharString\$ \\ CharString\$\,CharString \end{array} \right\} \\ > \text{%} \left\{ \begin{array}{l} SharedMemoryAddress \\ \$ \end{array} \right\} \end{array} \right]$$

BulkValues

$$: Buffer \; [\,,:StartIndex \; [\,,\;:NumberOfRows\,]\,]$$

Dynamic Parameter Substitution

$$(? \; [\; , \; \ldots \;] \;)$$

INSERT - 2

$$\texttt{INSERT INTO} \left\{ \begin{array}{l} [\;Owner.\;]\;TableName \\ [\;Owner.\;]\;ViewName \end{array} \right\} [\;(\;ColumnName\;[\;,\;\ldots\;]\;)\;] \quad QueryExpression$$

Labeled Statement

$$Label: \;\; Statement$$

LOCK TABLE

$$\texttt{LOCK TABLE}\;[\;Owner.\;]\;TableName\;\texttt{IN}\;\left\{ \begin{array}{l} \texttt{SHARE}\;[\;\texttt{UPDATE}\;] \\ \texttt{EXCLUSIVE} \end{array} \right\}\texttt{MODE}$$

LOG COMMENT

$$\texttt{LOG COMMENT}\;\left\{ \begin{array}{l} 'String' \\ :HostVariable \\ :ProcedureParameter \\ :ProcedureLocalVariable \\ ? \end{array} \right\}$$

OPEN

$$\texttt{OPEN}\;CursorName\;\left[\;\texttt{KEEP CURSOR}\;\left[\begin{array}{l} \texttt{WITH LOCKS} \\ \texttt{WITH NOLOCKS} \end{array} \right]\;\right]$$

$$\left[\;\texttt{USING}\;\left\{ \begin{array}{l} [\;\texttt{SQL}\;]\texttt{DESCRIPTOR}\left\{ \begin{array}{l} \texttt{SQLDA} \\ AreaName \end{array} \right\} \\ HostVariableName\;[\;[\;\texttt{INDICATOR}\;]:IndicatorVariable\;]\;[\;,\;\ldots\;] \end{array} \right\}\;\right]$$

PREPARE

$$\texttt{PREPARE}\;[\;\texttt{REPEAT}\;]\left\{ \begin{array}{l} StatementName \\ [\;Owner.\;]\;ModuleName\;[\;(SectionNumber)\;] \end{array} \right\}$$

$$[\;\texttt{IN}\;DBEFileSetName\;]\texttt{FROM}\;\left\{ \begin{array}{l} 'String' \\ :HostVariable \end{array} \right\}$$

PRINT

$$\texttt{PRINT}\;\left\{ \begin{array}{l} 'Constant' \\ :LocalVariable \\ :Parameter \\ ::Built\text{-}inVariable \end{array} \right\};$$

RAISE ERROR

RAISE ERROR $[ErrorNumber][$MESSAGE $ErrorText]$

REFETCH

REFETCH $CursorName$ INTO $\{\ :HostVariable\ [[$INDICATOR$]:Indicator]\ \}[\ ,\ ...\ ]$

RELEASE

RELEASE

REMOVE DBEFILE

REMOVE DBEFILE $DBEFileName$ FROM DBEFILESET $DBEFileSetName$

REMOVE FROM GROUP

REMOVE $\left\{ \begin{array}{l} DBEUserID \\ GroupName \\ ClassName \end{array} \right\} [\ ,\ ...\ ]$ FROM GROUP $[Owner.]TargetGroupName$

RESET

RESET $\left\{ \begin{array}{l} \text{SYSTEM.ACCOUNT}\ \left[\text{FOR USER}\ \left\{ \begin{array}{l} * \\ DBEUserID \end{array} \right\}\right] \\ \text{SYSTEM.COUNTER} \end{array} \right\}$

RETURN

RETURN $[ReturnStatus]$;

REVOKE

Revoke Authority on a Table or View

$$\text{REVOKE} \left\{ \begin{array}{l} \text{ALL } [\text{PRIVILEGES}] \\ \left[\begin{array}{l} \text{SELECT} \\ \text{INSERT} \\ \text{DELETE} \\ \text{ALTER} \\ \text{INDEX} \\ \text{UPDATE } [\,(\{\textit{ColumnName}\}[\,,\,\ldots\,])\,] \\ \text{REFERENCES } [\,(\{\textit{ColumnName}\}[\,,\,\ldots\,])\,] \end{array} \right] \mid\,,\,\ldots\,\mid \end{array} \right\}$$

$$\text{ON} \left\{ \begin{array}{l} [\textit{Owner.}]\,\textit{TableName} \\ [\textit{Owner.}]\,\textit{ViewName} \end{array} \right\} \text{FROM} \left\{ \begin{array}{l} \textit{DBEUserID} \\ \textit{GroupName} \\ \textit{ClassName} \\ \text{PUBLIC} \end{array} \right\} [\,,\,\ldots\,]\,[\text{CASCADE}]$$

Revoke RUN or EXECUTE or Authority

$$\text{REVOKE} \left[\begin{array}{l} \text{RUN ON } [\textit{Owner.}]\,\textit{ModuleName} \\ \text{EXECUTE ON PROCEDURE } [\textit{Owner.}]\,\textit{ProcedureName} \end{array} \right] \text{FROM}$$

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} \textit{DBEUserID} \\ \textit{GroupName} \\ \textit{ClassName} \end{array} \right\} [\,,\,\ldots\,] \\ \text{PUBLIC} \end{array} \right\}$$

Revoke CONNECT, DBA, or RESOURCE Authority

$$\text{REVOKE} \left\{ \begin{array}{l} \text{CONNECT} \\ \text{DBA} \\ \text{RESOURCE} \end{array} \right\} \text{FROM} \left\{ \begin{array}{l} \textit{DBEUserID} \\ \textit{GroupName} \\ \textit{ClassName} \end{array} \right\} [\,,\,\ldots\,]$$

SQL Syntax—Revoke DBEFileSet Authority

$$\text{REVOKE} \left\{ \begin{array}{l} \text{SECTIONSPACE} \\ \text{TABLESPACE} \end{array} \right\} \mid\,,\,\ldots\,\mid \text{ ON DBEFILESET } \textit{DBEFileSetName} \text{ FROM}$$

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} \textit{DBEUserID} \\ \textit{GroupName} \\ \textit{ClassName} \end{array} \right\} [\,,\,\ldots\,] \\ \text{PUBLIC} \end{array} \right\}$$

ROLLBACK WORK

$$\text{ROLLBACK WORK} \left[\begin{array}{l} \text{TO} \left\{ \begin{array}{l} \textit{SavePointNumber} \\ :\textit{HostVariable} \\ :\textit{LocalVariable} \\ :\textit{ProcedureParameter} \end{array} \right\} \\ \text{RELEASE} \end{array} \right]$$

SAVEPOINT

$$\text{SAVEPOINT} \begin{bmatrix} :HostVariable \\ :LocalVariable \\ :ProcedureParameter \end{bmatrix}$$

SELECT

Select Statement Level

$$[\text{BULK}]\, QueryExpression \left[\text{ORDER BY} \left\{ ColumnID \begin{bmatrix} \text{ASC} \\ \text{DESC} \end{bmatrix} \right\} [\, , \, \dots \,] \right]$$

Subquery Level

$$(QueryExpression)$$

Query Expression Level

$$\left\{ \begin{matrix} QueryBlock \\ (QueryExpression) \end{matrix} \right\} \left[\text{UNION}\,[\text{ALL}] \left\{ \begin{matrix} QueryBlock \\ (QueryExpression) \end{matrix} \right\} \right] [\, \dots \,]$$

Query Block Level

$$\text{SELECT} \begin{bmatrix} \text{ALL} \\ \text{DISTINCT} \end{bmatrix} SelectList \; [\text{INTO}\; HostVariableSpecification]$$

$$\text{FROM}\; FromSpec\; [\, , \, \dots \,]$$

$$[\text{WHERE}\; SearchCondition1]$$

$$[\text{GROUP BY}\; GroupColumnList\,]$$

$$[\text{HAVING}\; SearchCondition2]$$

SelectList

$$\left\{ \begin{matrix} * \\ [\,Owner.\,]\,Table.* \\ CorrelationName.* \\ Expression \\ [\,[\,Owner.\,]\,Table.\,]\,ColumnName \\ CorrelationName.\,ColumnName \end{matrix} \right\} [\, , \, \dots \,]$$

HostVariableSpecification—With BULK Option

$$:Buffer\; [\, , :StartIndex\; [\, , :NumberOfRows\,]\,]$$

HostVariableSpecification—Without BULK Option

$$\{ :HostVariable\; [\,[\text{INDICATOR}]:Indicator\,] \} [\, , \, \dots \,]$$

FromSpec

$$\left\{ \begin{array}{l} TableSpec \\ (\,FromSpec) \\ FromSpec \text{ NATURAL} \left[\begin{array}{l} \text{INNER} \\ \text{LEFT } [\,\text{OUTER}\,] \\ \text{RIGHT } [\,\text{OUTER}\,] \end{array}\right] \text{JOIN} \left\{\begin{array}{l} TableSpec \\ (\,FromSpec) \end{array}\right\} \\ FromSpec \left[\begin{array}{l} \text{INNER} \\ \text{LEFT } [\,\text{OUTER}\,] \\ \text{RIGHT } [\,\text{OUTER}\,] \end{array}\right] \text{JOIN} \left\{\begin{array}{l} TableSpec \\ (\,FromSpec) \end{array}\right\} \left\{\begin{array}{l} \text{ON } SearchCondition3 \\ \text{USING } (\,ColumnList) \end{array}\right\} \end{array} \right\}$$

TableSpec

$$[\,Owner.\,]\,TableName\,[\,CorrelationName\,]$$

SET CONNECTION

$$\text{SET CONNECTION} \left\{\begin{array}{l} {}'\,ConnectionName{}' \\ :\,HostVariable \end{array}\right\}$$

SET CONSTRAINTS

$$\text{SET } ConstraintType\,[\,,\,\ldots\,]\,\text{CONSTRAINTS} \left\{\begin{array}{l} \text{DEFERRED} \\ \text{IMMEDIATE} \end{array}\right\}$$

SET DEFAULT DBEFILESET

$$\text{SET DEFAULT} \left\{\begin{array}{l} \text{SECTIONSPACE} \\ \text{TABLESPACE} \end{array}\right\} \text{TO DBEFILESET } DBEFileSetName \text{ FOR PUBLIC}$$

SET DML ATOMICITY

$$\text{SET DML ATOMICITY AT} \left\{\begin{array}{l} \text{ROW} \\ \text{STATEMENT} \end{array}\right\} \text{LEVEL}$$

SET MULTITRANSACTION

$$\text{SET MULTITRANSACTION} \left\{\begin{array}{l} \text{ON} \\ \text{OFF} \end{array}\right\}$$

SETOPT

$$\text{SETOPT} \left\{\begin{array}{l} \text{CLEAR} \\ \text{GENERAL} \left\{\begin{array}{l} ScanAccess \\ JoinAlgorithm \end{array}\right\} [\,,\,\ldots\,] \\ \text{BEGIN} \left\{\text{GENERAL} \left\{\begin{array}{l} ScanAccess \\ JoinAlgorithm \end{array}\right\}\right\} [\,;\,\ldots\,]\,\text{END} \end{array} \right\}$$

Scan Access

$$[\,\text{NO}\,]\left\{\begin{array}{l}\text{SERIALSCAN}\\\text{INDEXSCAN}\\\text{HASHSCAN}\\\text{SORTINDEX}\end{array}\right\}$$

Join Algorithm

$$[\,\text{NO}\,]\left\{\begin{array}{l}\text{NESTEDLOOP}\\\text{NLJ}\\\text{SORTMERGE}\\\text{SMJ}\end{array}\right\}$$

SET PRINTRULES

$$\text{SET PRINTRULES}\left[\begin{array}{l}\text{ON}\\\text{OFF}\end{array}\right]$$

SET SESSION

$$\text{SET SESSION}\left\{\begin{array}{l}\text{ISOLATION LEVEL}\left\{\begin{array}{l}\text{RR}\\\text{CS}\\\text{RC}\\\text{RU}\\\text{REPEATABLE READ}\\\text{SERIALIZABLE}\\\text{CURSOR STABILITY}\\\text{READ COMMITTED}\\\text{READ UNCOMMITTED}\\:HostVariable1\end{array}\right\}\\[2pt]\text{PRIORITY}\left\{\begin{array}{l}Priority\\:HostVariable2\end{array}\right\}\\[2pt]\text{LABEL}\left\{\begin{array}{l}\text{'}LabelString\text{'}\\:HostVariable3\end{array}\right\}\\[2pt]ConstraintType\;[\,,\;\ldots\,]\text{CONSTRAINTS}\left\{\begin{array}{l}\text{DEFERRED}\\\text{IMMEDIATE}\end{array}\right\}\\[2pt]\text{DML ATOMICITY AT}\left\{\begin{array}{l}\text{STATEMENT}\\\text{ROW}\end{array}\right\}\text{LEVEL}\\[2pt]\left[\left\{\begin{array}{l}\text{PARALLEL}\\\text{NO}\end{array}\right\}\right]\text{FILL}\end{array}\right\}[\,,\;\ldots\,]$$

SET TRANSACTION

$$\text{SET TRANSACTION} \left\{ \begin{array}{l} \text{ISOLATION LEVEL} \left\{ \begin{array}{l} \text{RR} \\ \text{CS} \\ \text{RC} \\ \text{RU} \\ \text{REPEATABLE READ} \\ \text{SERIALIZABLE} \\ \text{CURSOR STABILITY} \\ \text{READ COMMITTED} \\ \text{READ UNCOMMITTED} \\ : HostVariable1 \end{array} \right\} \\ \text{PRIORITY} \left\{ \begin{array}{l} Priority \\ : HostVariable2 \end{array} \right\} \\ \text{LABEL} \left\{ \begin{array}{l} 'LabelString' \\ : HostVariable3 \end{array} \right\} \\ ConstraintType \left[, \ldots \right] \text{CONSTRAINTS} \left\{ \begin{array}{l} \text{DEFERRED} \\ \text{IMMEDIATE} \end{array} \right\} \\ \text{DML ATOMICITY AT} \left\{ \begin{array}{l} \text{STATEMENT} \\ \text{ROW} \end{array} \right\} \text{LEVEL} \end{array} \right\} \left[, \ldots \right]$$

SET USER TIMEOUT

$$\text{SET USER TIMEOUT} \left[\text{TO} \right] \left\{ \begin{array}{l} \left\{ \begin{array}{l} TimeoutValue \\ :HostVariable \end{array} \right\} \left[\begin{array}{l} \text{SECONDS} \\ \text{MINUTES} \end{array} \right] \\ \text{DEFAULT} \\ \text{MAXIMUM} \end{array} \right\}$$

SQLEXPLAIN

$$\text{SQLEXPLAIN} : HostVariable$$

START DBE

$$\text{START DBE } 'DBEnvironmentName' \left[\text{AS } ConnectionName' \right] \left[\text{MULTI} \right]$$
$$\left[\begin{array}{l} \text{BUFFER} = (DataBufferPages, \ LogBufferPages) \\ \text{TRANSACTION} = MaxTransactions \\ \text{MAXIMUM TIMEOUT} = \left\{ \begin{array}{l} TimeoutValue \left[\begin{array}{l} \text{SECONDS} \\ \text{MINUTES} \end{array} \right] \\ \text{NONE} \end{array} \right\} \\ \text{DEFAULT TIMEOUT} = \left\{ \begin{array}{l} TimeoutValue \left[\begin{array}{l} \text{SECONDS} \\ \text{MINUTES} \end{array} \right] \\ \text{MAXIMUM} \end{array} \right\} \\ \text{RUN BLOCK} = ControlBlockPages \end{array} \right] \left[, \ldots \right]$$

START DBE NEW

```
START DBE 'DBEnvironmentName' [ AS 'ConnectionName' ] [ MULTI ] NEW
⎡ ⎧ DUAL  ⎫                                                    ⎤
⎢ ⎨ AUDIT ⎬ | ... | LOG                                       ⎥
⎢ ⎩      ⎭                                                    ⎥
⎢ BUFFER = (DataBufferPages, LogBufferPages)                  ⎥
⎢ LANG = LanguageName                                         ⎥
⎢ TRANSACTION = MaxTransactions                               ⎥
⎢                   ⎧ TimeoutValue ⎡ SECONDS ⎤ ⎫              ⎥
⎢ MAXIMUM TIMEOUT = ⎨              ⎣ MINUTES ⎦ ⎬              ⎥
⎢                   ⎩ NONE                     ⎭              ⎥
⎢                   ⎧ TimeoutValue ⎡ SECONDS ⎤ ⎫              ⎥
⎢ DEFAULT TIMEOUT = ⎨              ⎣ MINUTES ⎦ ⎬              ⎥
⎢                   ⎩ MAXIMUM                  ⎭              ⎥
⎢ RUN BLOCK = ControlBlockPages                               ⎥
⎢                     ⎧ DefaultPartitionNumber ⎫              ⎥
⎢ DEFAULT PARTITION = ⎨ NONE                    ⎬             ⎥
⎢                     ⎩                         ⎭   | , ... | ⎥
⎢                     ⎧ CommentPartitionNumber ⎫              ⎥
⎢ COMMENT PARTITION = ⎨ DEFAULT                 ⎬             ⎥
⎢                     ⎩ NONE                    ⎭              ⎥
⎢ MAXPARTITIONS = MaximumNumberOfPartitions                   ⎥
⎢ AUDIT NAME = 'AuditName'                                    ⎥
⎢ ⎧ COMMENT       ⎫                                           ⎥
⎢ ⎪ DATA          ⎪                                           ⎥
⎢ ⎪ DEFINITION    ⎪                                           ⎥
⎢ ⎨ STORAGE       ⎬ | ... | AUDIT ELEMENTS                    ⎥
⎢ ⎪ AUTHORIZATION ⎪                                           ⎥
⎢ ⎪ SECTION       ⎪                                           ⎥
⎢ ⎩ ALL           ⎭                                           ⎥
⎢ DBEFile0Definition                                          ⎥
⎣ DBELogDefinition                                            ⎦
```

DBEFile0Definition

```
DBEFILE0 DBEFILE DBEFile0ID
WITH PAGES = DBEFile0Size, NAME = 'SystemFileName1'
```

DBELogDefinition

```
LOG DBEFILE DBELog1ID [ AND DBELog2ID ]
WITH PAGES = DBELogSize, NAME = 'SystemFileName2' [ AND 'SystemFileName3' ]
```

START DBE NEWLOG

```
START DBE 'DBEnvironmentName' [AS 'ConnectionName'] [MULTI] NEWLOG
```

$$
\left[
\begin{array}{l}
\left\{ \begin{array}{l} \text{ARCHIVE} \\ \text{DUAL} \\ \text{AUDIT} \end{array} \right\} \mid \ldots \mid \text{LOG} \\
\text{BUFFER} = (DataBufferPages,\ LogBufferPages) \\
\text{TRANSACTION} = MaxTransactions \\
\text{MAXIMUM TIMEOUT} = \left\{ \begin{array}{l} TimeoutValue \left[\begin{array}{l}\text{SECONDS}\\\text{MINUTES}\end{array}\right] \\ \text{NONE} \end{array} \right\} \\
\text{DEFAULT TIMEOUT} = \left\{ \begin{array}{l} TimeoutValue \left[\begin{array}{l}\text{SECONDS}\\\text{MINUTES}\end{array}\right] \\ \text{MAXIMUM} \end{array} \right\} \\
\text{RUN BLOCK} = ControlBlockPages \\
\text{DEFAULT PARTITION} = \left\{ \begin{array}{l} DefaultPartitionNumber \\ \text{NONE} \end{array} \right\} \\
\text{COMMENT PARTITION} = \left\{ \begin{array}{l} CommentPartitionNumber \\ \text{DEFAULT} \\ \text{NONE} \end{array} \right\} \\
\text{MAXPARTITIONS} = MaximumNumberOfPartitions \\
\text{AUDIT NAME} = 'AuditName' \\
\left\{ \begin{array}{l} \text{COMMENT} \\ \text{DATA} \\ \text{DEFINITION} \\ \text{STORAGE} \\ \text{AUTHORIZATION} \\ \text{SECTION} \\ \text{ALL} \end{array} \right\} \mid \ldots \mid \text{AUDIT ELEMENTS}
\end{array}
\right]
\mid ,\ \ldots \mid NewLogDefinition
$$

NewLogDefinition

```
LOG DBEFILE DBELog1ID [AND DBELog2ID]
WITH PAGES = DBELogSize, NAME = 'SystemFileName1'
[AND 'SystemFileName2']
```

STOP DBE

```
STOP DBE
```

TERMINATE USER

$$
\text{TERMINATE USER} \left\{ \begin{array}{l} DBEUserID \\ SessionID \end{array} \right\}
$$

TRANSFER OWNERSHIP

$$\text{TRANSFER OWNERSHIP OF} \left\{ \begin{array}{l} [\text{TABLE}]\,[\,Owner.\,]\,TableName \\ [\text{VIEW}]\,[\,Owner.\,]\,ViewName \\ \text{PROCEDURE}\,[\,Owner.\,]\,ProcedureName \\ \text{GROUP}\;GroupName \end{array} \right\} \text{TO}\;NewOwnerName$$

TRUNCATE TABLE

$$\text{TRUNCATE TABLE}\;[\,Owner.\,]\,TableName$$

UPDATE

$$\text{UPDATE} \left\{ \begin{array}{l} [\,Owner.\,]\,TableName \\ [\,Owner.\,]\,ViewName \end{array} \right\}$$

$$\text{SET} \left\{ ColumnName = \left\{ \begin{array}{l} Expression \\ \text{'}LongColumnIOString\text{'} \\ \text{NULL} \end{array} \right\} \right\} [\,,\,\ldots\,]$$

$$[\,\text{WHERE}\;SearchCondition\,]$$

LongColumnIOString

$$\left\{ \begin{array}{l} \left[< \left\{ \begin{array}{l} [\,PathName\mathbf{/}\,]\,FileName \\ \text{\%}\;SharedMemoryAddress \end{array} \right\} \right] \\ \\ \left[\left\{ \begin{array}{l} > \\ >> \\ >! \end{array} \right\} [\,PathName\mathbf{/}\,] \left\{ \begin{array}{l} FileName \\ CharString\$ \\ CharString\$\,CharString \end{array} \right\} \right] \\ \\ \left[> \text{\%} \left\{ \begin{array}{l} SharedMemoryAddress \\ \$ \end{array} \right\} \right] \end{array} \right\} |\,\ldots\,|$$

UPDATE STATISTICS

$$\text{UPDATE STATISTICS FOR TABLE} \left\{ \begin{array}{l} [\,Owner.\,]\,TableName \\ \text{SYSTEM}.\,SystemViewName \end{array} \right\}$$

UPDATE WHERE CURRENT

$$\text{UPDATE} \left\{ \begin{array}{l} [\,Owner.\,]\,TableName \\ [\,Owner.\,]\,ViewName \end{array} \right\}$$

$$\text{SET} \left\{ ColumnName = \left\{ \begin{array}{l} Expression \\ \text{'}LongColumnIOString\text{'} \\ \text{NULL} \end{array} \right\} \right\} [\,,\,\ldots\,]$$

$$\text{WHERE CURRENT OF}\;CursorName$$

LongColumnIOString

$$
\left\{
\begin{array}{l}
\left[< \left\{ \begin{array}{l} \left[\,PathName\textit{/}\,\right] FileName \\ \%\ SharedMemoryAddress \end{array} \right\} \right] \\[2em]
\left[\left\{ \begin{array}{l} > \\ >> \\ >! \end{array} \right\} \left[\,PathName\textit{/}\,\right] \left\{ \begin{array}{l} FileName \\ CharString\$ \\ CharString\$CharString \end{array} \right\} \right] \\[3em]
\left[> \% \left\{ \begin{array}{l} SharedMemoryAddress \\ \$ \end{array} \right\} \right]
\end{array}
\right\} |\ \ldots\ |
$$

VALIDATE

$$
\text{VALIDATE}\ \left[\,\text{DROP SETOPTINFO}\,\right]
$$
$$
\left\{
\begin{array}{l}
\text{MODULE}\ \left\{ \left[\,Owner.\,\right] ModuleName \right\} \left[\,,\ \ldots\ \right] \\[1.5em]
\text{PROCEDURE}\ \left\{ \left[\,Owner.\,\right] ProcedureName \right\} \left[\,,\ \ldots\ \right] \\[1.5em]
\text{ALL}\ \left\{ \begin{array}{l} \text{MODULES} \\ \text{PROCEDURES} \end{array} \right\}
\end{array}
\right\}
$$

WHENEVER

$$
\text{WHENEVER}\ \left\{ \begin{array}{l} \text{SQLERROR} \\ \text{SQLWARNING} \\ \text{NOT FOUND} \end{array} \right\} \left\{ \begin{array}{l} \text{STOP} \\ \text{CONTINUE} \\ \text{GOTO}\ Label \\ \text{GO TO}\ Label \end{array} \right\}
$$

WHILE

$$
\text{WHILE}\ Condition\ \text{DO}\ \left[\,Statement;\ \left[\ \ldots\ \right]\,\right] \text{ENDWHILE};
$$

ISQL Syntax Summary

CHANGE

C$\big[$HANGE$\big]$ *Delimiter OldString Delimiter NewString Delimiter* $\big[$@$\big]$

DO

DO $\begin{bmatrix} CommandNumber \\ CommandString \end{bmatrix}$

EDIT

ED$\big[$IT$\big]$ $\big[$*FileName*$\big]$

END

EN$\big[$D$\big]$

ERASE

ER$\big[$ASE$\big]$ *FileName*

EXIT

EX$\big[$IT$\big]$

EXTRACT

EXTRACT $\left\{ \begin{array}{l} \text{MODULE } \big[\mathit{Owner.}\big]\mathit{ModuleName} \big[\,,\,\ldots\,\big] \\ \text{SECTION } \big[\mathit{Owner.}\big]\mathit{ModuleName}(\mathit{SectionNumber}) \big[\,,\,\ldots\,\big] \\ \text{ALL MODULES} \end{array} \right\}$
$\big[$NO SETOPTINFO$\big]$ INTO *FileName*

HELP

$$\texttt{HE}\left[\texttt{LP}\right]\left\{\begin{array}{l} \texttt{@} \\ SQLStatement \\ ISQLCommand \end{array}\right\}\left[\begin{array}{l} \texttt{D}\left[\texttt{ESCRIPTION}\right] \\ \texttt{S}\left[\texttt{YNTAX}\right] \\ \texttt{E}\left[\texttt{AMPLE}\right] \end{array}\right]$$

HOLD

$$\texttt{HO}\left[\texttt{LD}\right]\left\{\begin{array}{l} SQLStatement \\ ISQLCommand \end{array}\right\}\left[EscapeCharacter;\left\{\begin{array}{l} SQLStatement \\ ISQLCommand \end{array}\right\}\right]\left[\ \ldots\ \right]$$

INFO

$$\texttt{IN}\left[\texttt{FO}\right]\left\{\begin{array}{l} \left[\ Owner.\ \right]TableName \\ \left[\ Owner.\ \right]ViewName \end{array}\right\}$$

INPUT

$$\texttt{INP}\left[\texttt{UT}\right]\left\{\begin{array}{l} \left[\ Owner.\ \right]TableName \\ \left[\ Owner.\ \right]ViewName \end{array}\right\}(\ ColumnName\ \left[\ ,ColumnName\right]\left[\ \ldots\ \right]\ )$$

$$\left\{\ (\ Value\ \left[\ ,Value\right]\left[\ \ldots\ \right]\ )\left[\begin{array}{l} \texttt{ROLLBACK WORK} \\ \texttt{COMMIT WORK} \end{array}\right]\right\}\left[\ \ldots\ \right]\ \texttt{E}\left[\texttt{ND}\right]$$

INSTALL

$$\texttt{IN}\left[\texttt{STALL}\right]FileName\ \left[\texttt{DROP}\right]\left[\texttt{IN}\ DBEFileSetName\right]\left[\texttt{NO OPTINFO}\right]$$

LIST FILE

$$\texttt{LI}\left[\texttt{ST}\right]\texttt{F}\left[\texttt{ILE}\right]FileName$$

LIST HISTORY

$$\texttt{LI}\left[\texttt{ST}\right]\texttt{H}\left[\texttt{ISTORY}\right]\left\{\begin{array}{l} CommandNumber \\ \texttt{@} \end{array}\right\}$$

LIST INSTALL

$$\texttt{LI}\left[\texttt{ST}\right]\texttt{I}\left[\texttt{NSTALL}\right]FileName$$

LIST SET

$$\text{LI}\left[\text{ST}\right]\text{S}\left[\text{ET}\right]\left\{\begin{array}{l}Option\\ \texttt{@}\end{array}\right\}$$

LOAD

$$\text{LO}\left[\text{AD}\right]\left[\text{P}\left[\text{ARTIAL}\right]\right]\text{FROM}\left\{\begin{array}{l}\text{E}\left[\text{XTERNAL}\right]\\ \text{I}\left[\text{NTERNAL}\right]\end{array}\right\}InputFileName\;\left[\text{AT}\;StartingRow\right]$$

$$\left[\text{FOR}\;NumberOfRows\right]\text{TO}\left\{\begin{array}{l}\left[Owner.\right]TableName\\ \left[Owner.\right]ViewName\end{array}\right\}\left[\begin{array}{l}ExternalInputSpec\\ \text{USING}\;DescriptionFileName\end{array}\right]$$

$$\left\{\begin{array}{l}\text{Y}\left[\text{ES}\right]PatternLocation\;Pattern\\ \text{N}\left[\text{O}\right]\end{array}\right\}$$

ExternalInputSpec

$$\left\{\begin{array}{l}ColumnName\;StartingLocation\;Length\left[NullIndicator\right]\\ \left[FormatType\right]\end{array}\right\}\left[\;\ldots\;\right]\text{E}\left[\text{ND}\right]$$

RECALL

$$\text{REC}\left[\text{ALL}\right]\left\{\begin{array}{l}\text{C}\left[\text{URRENT}\right]\\ \text{F}\left[\text{ILE}\right]FileName\\ \text{H}\left[\text{ISTORY}\right]CommandNumber\end{array}\right\}$$

REDO

$$\text{RED}\left[\text{O}\right]\left[\begin{array}{l}CommandNumber\\ CommandString\end{array}\right]$$

Subcommands

B	Break
D	Delete
E	Exit
H	Help
I	Insert
L	List
R	Replace
X	Execute
+[n]	Forward n
-[n]	Backward n
Return	Next Line

RENAME

REN[AME] *OldFileName NewFileName*

SELECTSTATEMENT

SelectStatement; [PA[USE];] [*BrowseOption*;] [...]E[ND]

SET

SE[T] *Option OptionValue*

Options and Values

```
AUTOC[OMMIT] ON | OFF
AUTOL[OCK] ON | OFF
AUTOS[AVE] NumberofRows
C[ONTINUE] ON | OFF
CONV[ERT] ASCII | EBCDIC | OFF
EC[HO] ON | OFF
ECHO_[ALL] ON | OFF
EDITOR EditorName
ES[CAPE] Character
EXIT[_ON_DBERR] ON | OFF
EXIT_ON_DBWARN ON | OFF
FL[AGGER] FlaggerName
F[RACTION] Length
N[ULL] [Character]
OU[TPUT] FileName
OW[NER] OwnerName
LOAD_B[UFFER] BufferSize
PA[GEWIDTH] PageWidth
PR[OMPT] PromptString
```

SQLGEN

SQLG[EN]

SQLUTIL

SQLU[TIL]

START

$$\text{STA}[\text{RT}]\,[\,CommandFileName\,]\,[\,(\,Value\,[\,,Value\,]\,[\,\ldots\,]\,)\,]$$

STORE

$$\text{STO}[\text{RE}]\,FileName\,[\,\text{R}[\text{EPLACE}]\,]$$

SYSTEM

$$\left\{ \begin{array}{l} \text{SY}[\text{STEM}] \\ ! \end{array} \right\}\,[\,HP\text{-}UXCommand\,]$$

UNLOAD

$$\text{U}[\text{NLOAD}]\,\text{TO}\,\left\{ \begin{array}{l} \text{E}[\text{XTERNAL}] \\ \text{I}[\text{NTERNAL}] \end{array} \right\}\,OutputFileName$$

$$\text{FROM}\,\left\{ \begin{array}{l} [\,\text{Owner.}\,]\text{TableName} \\ [\,\text{Owner.}\,]\text{ViewName} \\ "SelectCommand" \end{array} \right\}\,ExternalOutputSpec$$

ExternalOutputSpec

$$DescriptionFileName\,\left\{ \begin{array}{l} OutputLength\,[\,FractionLength\,] \\ [\,NullIndicator\,] \end{array} \right\}\,[\,\ldots\,]$$

Index

Special characters

@, 2-18

A

aborting commands, 2-5
application program
  using SQL statements, 3-15
application programmers, 1-2
authorization, 1-3
AUTOCOMMIT option, 4-53
  and INPUT, 2-4, 3-4, 4-22, 4-23
  and INSTALL, 2-4, 4-25
  and LOAD, 2-4, 3-15
AUTOLOCK option, 4-53
automatic DBE session termination, 2-3
automatic transaction management, 2-3
AUTOSAVE option, 4-53
  and INPUT, 3-4
  and LOAD, 3-15

B

background
  running ISQL in, 2-12
banner, ISQL, 2-1
beginning a DBE session, 2-2
BEGIN WORK
  automatically by ISQL, 2-3
  use of, 2-3
BINARY data
  in description file, 4-37
BOTTOM, 3-5

C

CHANGE
  and command buffer, 2-14
  and special characters, 2-7, 4-4
  syntax, 4-4
changing
  command history buffer contents, 2-14
  files, 4-8
changing commands
  with CHANGE, 4-4
  with REDO, 4-44
check constraint
  in CREATE TABLE, A-6

column definition, 4-20
column names
  and INFO, 3-8, 4-20
  in CREATE TABLE, A-5
columns
  maximum allowed in tables, A-5
command
  entry, 2-8
  errors, 2-22
  Manager, ISQL, 2-8
  router, 2-8
  sources, 2-8
  termination, 2-4
  terminator, 2-6
command buffer, 2-13–14
  and command files, 4-42
  and command history buffer, 2-13
  and HOLD, 4-18
  and parameters, 2-8
  displaying current contents of, 4-42
  editing contents of, 4-4
  executing commands in, 4-60
  filled from command history buffer, 4-42
  ISQL commands that fill, 4-42, 4-61
  ISQL commands that operate on, 4-43
  putting commands in, 4-18
  related ISQL commands, 2-13
  storing contents in a file, 4-64
command file, 2-9–11
  and EDIT, 2-10
  and parameters, 2-7–8
  and STORE, 2-9
  comments in, 2-10
  creating, 2-10, 4-61
  creating from command buffer, 4-64
  displaying, 4-27
  error messages, 2-11
  executing, 2-11, 4-61
  executing commands in, 4-60
  line limit, 2-10
  maintenance, 2-9
  pausing to review query result, 4-49
  SELECT in, 2-11
command history buffer, 2-14–16
  and LIST HISTORY, 4-28
  related ISQL commands, 2-14

U

UNLOAD, 4-67–74
  direct to tape, 4-70
  EXTERNAL option, 3-9, 3-11
  INTERNAL option, 3-11
  syntax, 4-67
  use of, 3-9
  using a script file, 4-74
unloading data
  and UNLOAD, 4-67
  from the command line, 3-10
  prompting mode, 3-10
  ways of, 3-10, 3-12
  with a command file, 3-10
unloading from tables, 4-67
UP, 3-5
UPDATE

  ISQL support of, 3-3
  updating files, 4-8
uses for ISQL, 1-2

V

VALIDATE, 4-26
values
  passing, 4-61
view definition, 3-8, 4-20

W

wildcard, 2-18

Z

ZONED DECIMAL
  converting, 4-38