CHAPTER 5

Using Interactive SQL

About this chapter

This chapter discusses how to run the Interactive SQL utility. Ideally, you should run the software on your computer as you work through this chapter.

This chapter presents the various facilities offered by the Interactive SQL environment.

Contents

Introduction to Interactive SQL

Interactive SQL is a utility for sending SQL statements to the database server. You can use it for the following purposes:

- Browsing the information in a database.
- Trying out SQL statements that you plan to include in an application.
- Loading data into a database, and carrying out other administrative tasks.

In addition, Interactive SQL can run command files. You can build repeatable scripts to run against a database and then use Interactive SQL to execute these scripts.

Starting Interactive SQL

The way you start Interactive SQL depends on your operating system.

- In Windows 95 or Windows NT, select Interactive SQL from the Adaptive Server Anywhere program group.
- In OS/2 or Windows 3.1, double-click Interactive SQL in the Adaptive Server Anywhere program group.
- To start Interactive SQL in UNIX type the following command at a system prompt:

  ```
  dbisql
  ```

- To start Interactive SQL on a NetWare machine, type the command:

  ```
  load dbisql.nlm
  ```

Entering commands

The **Command window** appears at the bottom of the Interactive SQL screen. It is a standard edit control for typing Interactive SQL commands. If more lines are typed than will fit in this window, the window automatically scrolls. You can scroll the window using the cursor keys or the scroll bar on the right side of the window. This window can also be made larger and maximized to full screen size in the standard fashion for the operating system.

Commands are executed by pressing the execute key (F9) or you can click EXECUTE.

Multiple commands can be entered at once by separating them with semicolons. Commands can be stored to an ASCII file or loaded from an ASCII file by choosing File➤Save or Open from the menu.

Command recall

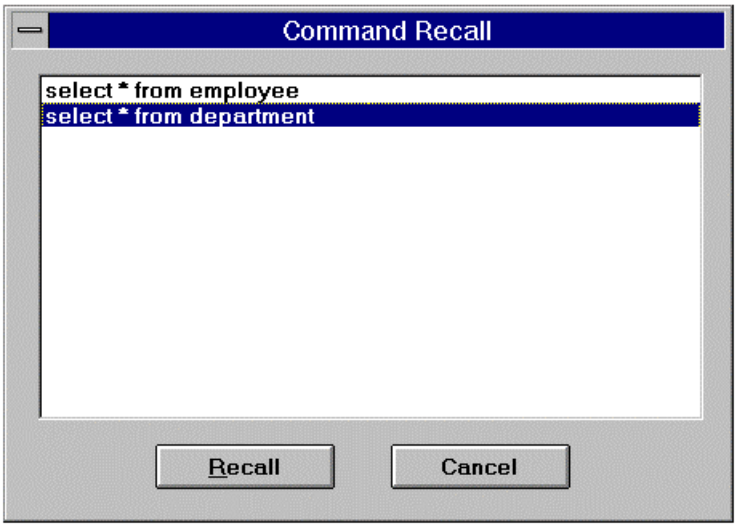
Before using Command Recall check the system tray to make sure the Standalone Database Server is running.

1 Type the following in the command window:

```
SELECT * FROM department
```

2 Press F9.

The contents of the **department** database table are displayed in the Data window. As you execute commands with Interactive SQL, they are saved in a command history. To recall commands, choose Command➤Recall from the menu bar. This activates the command recall window.



The command recall window displays the first line of the last 15 commands executed. Use the cursor up and down keys to scroll through the commands.

Position the cursor on the first command that you executed, which was:

```
SELECT *
FROM department
```

and press the ENTER key. The cursor returns to the command window with the selected command in it. You can now re-execute that command or modify it to make a new command.

More recall keys

The following keys can also be used to recall previous commands:

Key sequence	Description
CTRL+R	Brings up the command recall window
CTRL+P	Cycles backwards through previously executed commands. Retrieved commands are placed into the command window
CTRL+N	Cycles forward through previously executed commands

Canceling an Interactive SQL command

The STOP button is used to cancel a command.

A Stop operation stops current processing and prompts for the next command. If a command file was being processed, you are prompted for an action to take (Stop command file, Continue, or Exit Interactive SQL). These actions can be controlled with the Interactive SQL ON_ERROR option (see "ON_ERROR option" on page 165 of the book *Adaptive Server Anywhere Reference Manual*).

Reported errors

When an abort is detected, one of three different errors will be reported depending upon when the abort is detected.

1. If the abort is detected when Interactive SQL is processing the request (as opposed to the database engine), then the following message is displayed:

   ISQL command terminated by user

   Interactive SQL stops processing immediately and the database transaction is left alone.

2. If the cancel is detected by the database engine while processing a standard data manipulation command (SELECT, INSERT, DELETE, and UPDATE) and the engine is not running in bulk operations mode, then the following message is displayed.

   Statement interrupted by user.

   The effects of the current command are undone, but the rest of the transaction is left intact.

3. If the abort is detected while the database engine is processing a data definition command (CREATE, DROP, ALTER, etc.), the following message appears:

   Terminated by user -- transaction rolled back

   Since data definition commands all perform a COMMIT automatically before the command starts, the effect of the ROLLBACK is to just cancel the current command.

   This message also occurs when the database engine is running in bulk operations mode executing a command that modifies the database (INSERT, UPDATE, and DELETE). In this case, ROLLBACK cancels not only the current command, but everything that has been done since the last COMMIT. In some cases, it may take a considerable amount of time for the database engine to perform the automatic ROLLBACK.

Displaying data

One of the principal uses of Interactive SQL is to look at information in databases.

The database used in this tutorial is for a fictional company. The sample database contains information about employees, departments, sales orders, and so on.

All this information is organized into a number of **tables**, consisting of **rows** and **columns**.

You display information from a database using the SELECT statement. The following example shows the command to type in the Interactive SQL command window. Once you have typed the command, you must click Execute to carry out the command. The example displays the first several columns and rows of the results of the query, which are displayed in the Interactive SQL data window. The format is used throughout this manual.

❖ **To list all the columns and rows of the employee table:**

- Type the following:

```
SELECT *
FROM employee
```

emp_id	manager_id	emp_lname	emp_fname	...
102	501	Fran	Whitney	...
105	501	Matthew	Cobb	...
129	902	Philip	Chin	...
148	1293	Julie	Jordan	...
160	501	Robert	Breault	...
...				

Notes

- SQL statements are case insensitive. SELECT is the same as select is the same as Select. In the examples, SQL keywords are shown in upper case, but you do not have to type them in upper case.
- SQL statements can be typed on more than one line. You can type the statements all on one line, or break them by hitting enter at the end of each line. Some SQL statements, such as the SELECT statement, consist of several parts, called **clauses**. In many examples, each clause is placed on a separate line, but you do not have to type them this way.

The Interactive SQL Data window displays a set of rows and columns containing information about the employees. Each row contains information about one employee, and each column contains a particular piece of information for all employees.

Scrolling the data window

When you type the command

```
SELECT * FROM employee
```

in the Interactive SQL command window, the visible portion of the Interactive SQL data window cannot hold the entire **employee** table.

The visible portion of the data window does not display all the information about each employee, and does not display the entire list of employees.

Viewing other columns

To see more information about each employee (that is, other columns) you use the scroll bar at the bottom of the data window. This is a standard Windows or OS/2 scroll bar.

Viewing other rows

To see more information on other employees (that is, other rows), use the scroll bar to the right of the data window. The employee table in the sample database has information on about 75 employees.

Sometimes, the vertical scroll bar behaves slightly differently than standard scroll bars, as the number of rows in the result may be unknown. In this case, a guess as to the number of rows is used. If Interactive SQL determines that its guess is wrong, the guess is adjusted and the slider "jumps".

Function keys

Interactive SQL uses some function keys and special keys as follows:

Function key	Description
F1	Help
F5	Move data to the left by one column in the data window
SHIFT+F5	Move data to the left by one character
F6	Move data to the right by one column
SHIFT+F6	Move data to the right by one character
F7	Display a list of the tables in the database. The cursor up and down keys can be used to scroll through the table names changing the highlighted table name. With the list displayed, pressing ENTER will insert the current table name into the command window at the cursor position. The F7 key can be used while the table list is displayed, and a list of columns will be displayed for the highlighted table. Again, ENTER can be used to select the highlighted column name and put it into the command window at the cursor position.
F9	Execute the command that is in the command window. This operation can also be performed with the mouse by clicking EXECUTE.
F10	Activate the menus
PGUP	Move data up a page
PGDN	Move data down a page
CTRL+PGUP	Move to top of data
CTRL+PGDN	Move to bottom of data

Function keys for UNIX Interactive SQL

Interactive SQL on UNIX may be run on a variety of terminals. Some terminals prevent the Interactive SQL utility from distinguishing the ALT and ESC keys, and the function keys. This section describes alternatives in these situations.

- **ALT key** On terminals for which the ALT key is not supported in Interactive SQL, to enter an ALT- key, use CTRL-A followed by the key to which the ALT modifier is to be applied. For example, to see the File menu, type CTRL-A F.

If you are using Interactive SQL from a remote terminal such as a vt100 emulator you may be able to configure the terminal to use **emacs mode** so that it sends ALT-*key* as ESC *key*. Interactive SQL also recognizes these escape sequences. You can type ESC *key* yourself but they must be typed in quick succession otherwise the keys are interpreted individually.

- **Function keys** On terminals for which function keys do not appear to be supported in Interactive SQL, you can use CTRL-F followed by a single digit for the function key number. For example, you can enter F4 by typing CTRL-F 4.
- **Shift and Ctrl keys** If necessary, key sequences can also be defined for SHIFT and CTRL to be applied to the next key (so that SHIFT-*FunctionKey* or CTRL-PGDOWN can be entered). These sequences are controlled by a terminfo extension (*tix*) file.

  Interactive SQL first looks for *${TERM}.tix* in *${HOME}${ASANY} / tix*, *${ASANY} /bin*, and then throughout *PATH*. If *$TERM.tix* is not found, Interactive SQL searches for *default.tix* in the same directories.

Running command files

This section describes how to use the Interactive SQL command window to enter multiple commands at a time and how to process files consisting of a set of commands.

Entering multiple statements

SQL statements can get quite large. You have already seen how to use the editor to enter statements on several lines. The Interactive SQL environment also allows multiple commands to be entered at the same time. This is done by ending each statement with a semi-colon (;).

Example: entering multiple statements

❖ **To enter multiple statement in the Interactive SQL Command window**

1 Try entering the following three commands into the Command window.

```
UPDATE employee
SET dept_id = 400,
    manager_id = 1576
WHERE emp_id = 467;

UPDATE employee
SET dept_id = 400,
    manager_id = 1576
WHERE emp_id = 195;

SELECT *
FROM employee
WHERE emp_id IN ( 195, 467 )
```

2 Press the execute key (F9). All three statements are executed. After execution, the commands are left in the Command window.

Saving statements as command files

You can save commands you enter in Interactive SQL to a command file. This keeps a permanent record of the SQL commands so they can be used later if you wish.

❖ **To save statements as a command file:**

1 Choose File ➤Save As from the menu bar. You are then prompted for a filename.

2 Type a file name (for example, *transfer.sql*) and press ENTER.

3 The command file can be run using the Interactive SQL READ command, but you should rollback the changes first. Press the ESCAPE key to clear the editor and then execute the ROLLBACK WORK command.

4 Now enter the following command:

```
READ transfer.sql
```

This command executes the command file *transfer.sql* which contains the three commands that we saved previously. As each command is executed, it flashes up in the Command window.

What are command files?

Command files are just ASCII files containing SQL statements as you see them in the editor. You can use any editor you like to create command files. You can include comment lines along with the SQL statements to be executed. Command files are also commonly called **scripts**.

Executing command files

You can execute command files in the following ways:

- You can use the Interactive SQL READ command to execute command files.

  The following statement executes the file *temp.sql*:

  ```
  READ temp.sql
  ```

- You can load a command file into the Interactive SQL Command window and execute it directly from there.

  You load command files back into the Command window by choosing File ➤Open. Enter *transfer.sql* when prompted for the file name.

- You can supply a command file as a command line argument for Interactive SQL.

The Command window in Interactive SQL has a limit of 500 lines. For command files larger than this, you should use a generic editor capable of handling large files. The READ command has no limit on the number of lines that can be read.

Command files with parameters

An example of a command file that would take a parameter is a command file to show the department an employee belongs to, providing the employee's name as a parameter.

❖ **To create a command file with parameters:**

1 Create a command file as listed below.
The PARAMETERS command is used to give names to the parameters passed to a command file. In this case, we are giving the first parameter the name **employee_name**. The parameters are then used in the rest of the command file by enclosing them in braces ({}). Save the command file to **emp_dept.sql**.

```
parameters employee_name;

SELECT emp_lname, dept_name
FROM employee
    NATURAL JOIN department
WHERE emp_lname = {employee_name};
```

2 Run this command file by typing:

```
READ emp_dept.sql
```

3 You will be prompted for the **employee_name**. Enter the following value, including the single quotes:

```
'Whitney'
```

You should now see that the employee with surname **Whitney** is in the R&D department.

Parameters specifies on the READ command

Parameters can also be specified on the READ command. Try the following command:

```
READ emp_dept.sql 'Whitney'
```

In this case you have specified the parameter on the READ command, so Interactive SQL will not prompt for it. Interactive SQL will only prompt for parameters that are named in the PARAMETERS command but are not supplied on the READ command.

Note:

- The Parameters command is only allowed in command files and cannot be executed from the command window.