

Overview of SQL*Plus

You can use the SQL*Plus program in conjunction with the SQL database language and its procedural language extension, PL/SQL. The SQL database language allows you to store and retrieve data in Oracle. PL/SQL allows you to link several SQL commands through procedural logic.

SQL*Plus enables you to manipulate SQL commands and PL/SQL blocks, and to perform many additional tasks as well. Through SQL*Plus, you can

- enter, edit, store, retrieve, and run SQL commands and PL/SQL blocks
- format, perform calculations on, store, and print query results in the form of reports
- list column definitions for any table
- access and copy data between SQL databases
- send messages to and accept responses from an end user
- perform database administration

Basic Concepts

The following definitions explain concepts central to SQL*Plus:

command	An instruction you give SQL*Plus or Oracle.
block	A group of SQL and PL/SQL commands related to one another through procedural logic.
table	The basic unit of storage in Oracle.
query	A SQL command (specifically, a SQL SELECT command) that retrieves information from one or more tables.
query results	The data retrieved by a query.
report	Query results formatted by you through SQL*Plus commands.

Who Can Use SQL*Plus ?

The SQL*Plus, SQL, and PL/SQL command languages are powerful enough to serve the needs of users with some database experience, yet straightforward enough for new users who are just learning to work with Oracle.

The design of the SQL*Plus command language makes it easy to use. For example, to give a column labelled ENAME in the database the clearer heading "Employee", you might enter the following command:

```
COLUMN ENAME HEADING EMPLOYEE
```

Similarly, to list the column definitions for a table called EMP, you might enter this command:

```
DESCRIBE EMP
```

Other Ways of Working with Oracle

Oracle tools for Network Computing Architecture help developers to productively and economically build, manage and deploy high-performance and robust enterprise applications for Network Computing.

JDeveloper Suite	a 3GL development tool for building component based, server-centric applications in Java
Oracle Enterprise Developer Suite	an integrated and flexible set of tools for building enterprise-class database applications for client/server and the web
Oracle Application Server	a tool which enables database access through web browsers and the Internet
Oracle Designer	a set of client/server design tools for database applications
Oracle Developer	a set of client/server and web development tools
Oracle Discoverer	a set of end-user query tools
Oracle Programmer	a set of 3GL programming language interfaces
Oracle Reports	a publishing and reporting solution to disseminate dynamic information across corporate intranets or on the Internet
Oracle Workflow	a complete workflow management system that supports business process definition and automation
Oracle Express	a powerful OnLine Analytical Processing (OLAP) server, tools, and pre-built applications for financial, and sales and marketing analysis
Oracle Media Objects	a development tool for object-oriented multimedia applications
Oracle Mobile Agents	a tool for applications using mobile and/or detached clients

What You Need to Run SQL*Plus

To run SQL*Plus, you need hardware, software, operating system specific information, a username and password, and access to one or more tables.

Hardware and Software

Oracle and SQL*Plus can run on many different kinds of computers. Your computer's operating system manages the computer's resources and mediates between the computer hardware and programs such as SQL*Plus. Different computers use different operating systems. For information about your computer's operating system, see the documentation provided with the computer.

Before you can begin using SQL*Plus, both Oracle and SQL*Plus must be installed on your computer. Note that in order to take full advantage of the enhancements in SQL*Plus release 8.1.5, you must have Oracle8i. For a list of SQL*Plus release 8.1.5 enhancements, see [Appendix B](#).

If you have multiple users on your computer, your organization should have a Database Administrator (called a DBA) who supervises the use of Oracle.

The DBA is responsible for installing Oracle and SQL*Plus on your system. If you are acting as DBA, see the instructions for installing Oracle and SQL*Plus in the Oracle installation and user's manual(s) provided for your operating system.

Information Specific to Your Operating System

A few aspects of Oracle and SQL*Plus differ from one type of host computer and operating system to another. These topics are discussed in the Oracle installation and user's manual(s), published in a separate version for each host computer and operating system that SQL*Plus supports.

Keep a copy of your Oracle installation and user's manual(s) available for reference as you work through this Guide. When necessary, this Guide will refer you to your installation and user's manual(s).

Username and Password

When you start SQL*Plus, you will need a *username* that identifies you as an authorized Oracle user and a *password* that proves you are the legitimate owner of your username. See the PASSWORD command in [Chapter 8](#) for details on how to change your password. The demonstration username, SCOTT, and password, TIGER, may be set up on your system during the installation procedure. In this case, you can use the Oracle username SCOTT and password TIGER with the EMP and DEPT tables ([Figure 1-1](#) and [Figure 1-2](#)).

Multi-User Systems

If several people share your computer's operating system, your DBA can set up your SQL*Plus username and password. You will also need a system username and password to gain admittance to the operating system. These may or may not be the same ones you use with SQL*Plus.

Single-User Systems

If only one person at a time uses your computer, you may be expected to perform the DBAs functions for yourself. In that case, you can use the Oracle username SCOTT and password TIGER. If you want to define your own username and password, see the [*Oracle8i SQL Reference*](#).

Access to Sample Tables

Each table in the database is "owned" by a particular user. You may wish to have your own copies of the sample tables to use as you try the examples in this Guide. To get your own copies of the tables, see your DBA or run the Oracle-supplied command file named DEMOBLD (you run this file from your operating system, not from SQL*Plus).

When you have no more use for the sample tables, remove them by running another Oracle-supplied command file named DEMODROP. For instructions on how to run DEMOBLD and DEMODROP, see the Oracle installation and user's manual(s) provided for your operating system.

SQL Commands

@	Runs the specified command file.
@@	Runs a command file.
/	Executes the SQL command or PL/SQL block.
ACCEPT	Reads a line of input and stores it in a given user variable.
APPEND	Adds specified text to the end of the current line in the buffer.
ARCHIVE LOG	Starts or stops the automatic archiving of online redo log files, manually (explicitly) archives specified redo log files, or displays information about redo log files.
ATTRIBUTE	Specifies display characteristics for a given attribute of an Object Type column, and lists the current display characteristics for a single attribute or all attributes.
BREAK	Specifies where and how formatting will change in a report, or lists the current break definition.
BTITLE	Places and formats a specified title at the bottom of each report page, or lists the current BTITLE definition.
CHANGE	Changes text on the current line in the buffer.
CLEAR	Resets or erases the current clause or setting for the specified option, such as BREAKS or COLUMNS.
COLUMN	Specifies display characteristics for a given column, or lists the current display characteristics for a single column or for all columns.
COMPUTE	Calculates and prints summary lines, using various standard computations, on subsets of selected rows, or lists all COMPUTE definitions.
CONNECT	Connects a given username to Oracle.
COPY	Copies data from a query to a table in a local or remote database.
DEFINE	Specifies a user variable and assigns it a CHAR value, or lists the value and variable type of a single variable or all variables.
DEL	Deletes one or more lines of the buffer.

DESCRIBE	Lists the column definitions for the specified table, view, or synonym or the specifications for the specified function or procedure.
DISCONNECT	Commits pending changes to the database and logs the current username off Oracle, but does not exit SQL*Plus.
EDIT	Invokes a host operating system text editor on the contents of the specified file or on the contents of the buffer.
EXECUTE	Executes a single PL/SQL statement.
EXIT	Terminates SQL*Plus and returns control to the operating system.
GET	Loads a host operating system file into the SQL buffer.
HELP	Accesses the SQL*Plus help system.
HOST	Executes a host operating system command without leaving SQL*Plus.
INPUT	Adds one or more new lines after the current line in the buffer.
LIST	Lists one or more lines of the SQL buffer.
PASSWORD	Allows a password to be changed without echoing the password on an input device.
PAUSE	Displays an empty line followed by a line containing text, then waits for the user to press [Return], or displays two empty lines and waits for the user's response.
PRINT	Displays the current value of a bind variable.
PROMPT	Sends the specified message or a blank line to the user's screen.
RECOVER	Performs media recovery on one or more table spaces, one or more data files, or the entire database.
REMARK	Begins a comment in a command file.
REPFOOTER	Places and formats a specified report footer at the bottom of each report, or lists the current REPFOOTER definition.
REPHEADER	Places and formats a specified report header at the top of each report, or lists the current REPHEADER definition.

RUN	Lists and executes the SQL command or PL/SQL block currently stored in the SQL buffer.
SAVE	Saves the contents of the SQL buffer in a host operating system file (a command file).
SET	Sets a system variable to alter the SQL*Plus environment for your current session.
SHOW	Shows the value of a SQL*Plus system variable or the current SQL*Plus environment.
SHUTDOWN	Shuts down a currently running Oracle instance, optionally closing and dismounting the database.
SPOOL	Stores query results in an operating system file and, optionally, sends the file to a printer.
START	Executes the contents of the specified command file.
STARTUP	Starts an Oracle instance with several options, including mounting and opening a database.
STORE	Saves attributes of the current SQL*Plus environment in a host operating system file (a command file).
TIMING	Records timing data for an elapsed period of time, lists the current timer's title and timing data, or lists the number of active timers.
TTITLE	Places and formats a specified title at the top of each report page, or lists the current TTITLE definition.
UNDEFINE	Deletes one or more user variables that you defined either explicitly (with the DEFINE command) or implicitly (with an argument to the START command).
VARIABLE	Declares a bind variable that can be referenced in PL/SQL.
WHENEVER OSERROR	Exits SQL*Plus if an operating system command generates an error.
WHENEVER SQLERROR	Exits SQL*Plus if a SQL command or PL/SQL block generates an error.

@ ("at" sign)

Purpose

Runs the specified command file.

Syntax

@ *file_name*[.ext] [*arg...*]

Terms and Clauses

Refer to the following list for a description of each term or clause:

***file_name*[.ext]**

Represents the command file you wish to run. If you omit *ext*, SQL*Plus assumes the default command-file extension (normally SQL). For information on changing the default extension, see the SUFFIX variable of the SET command in this chapter.

When you enter @ *file_name.ext*, SQL*Plus searches for a file with the filename and extension you specify in the current default directory. If SQL*Plus does not find such a file, SQL*Plus will search a system-dependent path to find the file. Some operating systems may not support the path search. Consult the Oracle installation and user's manual(s) provided for your operating system for specific information related to your operating system environment.

arg...

Represent data items you wish to pass to parameters in the command file. If you enter one or more arguments, SQL*Plus substitutes the values into the parameters (&1, &2, and so forth) in the command file. The first argument replaces each occurrence of &1, the second replaces each occurrence of &2, and so forth.

The @ command DEFINES the parameters with the values of the arguments; if you run the command file again in this session, you can enter new arguments or omit the arguments to use the current values.

For more information on using parameters, refer to the subsection "Passing Parameters through the START Command" under "Writing Interactive Commands" in Chapter 3.

Usage Notes

You can include in a command file any command you would normally enter interactively (typically, SQL, SQL*Plus commands, or PL/SQL blocks).

An EXIT or QUIT command used in a command file terminates SQL*Plus.

The @ command functions similarly to START.

If the START command is disabled (see "Disabling SQL*Plus, SQL, and PL/SQL Commands" in Appendix E), this will also disable the @ command. See START in this chapter for information on the START command.

SQL*Plus removes the SQLTERMINATOR (a semicolon by default) before the @ command is issued. A workaround for this is to add another SQLTERMINATOR. See the SQLTERMINATOR variable of the SET command in this chapter for more information.

Examples

To run a command file named PRINTRPT with the extension SQL, enter

```
SQL> @PRINTRPT
```

To run a command file named WKRPT with the extension QRY, enter

```
SQL> @WKRPRT.QRY
```

@@ (double "at" sign)

Purpose

Runs a command file. This command is identical to the @ ("at" sign) command except that it looks for the specified command file in the same path as the command file from which it was called.

Syntax

```
@@ file_name[.ext]
```


Terms and Clauses

Refer to the following for a description of the term or clause:

file_name[.ext]

Represents the nested command file you wish to run. If you omit *ext*, SQL*Plus assumes the default command-file extension (normally SQL). For information on changing the default extension, see the SUFFIX variable of the SET command in this chapter.

When you enter *@@file_name.ext* from within a command file, SQL*Plus runs *file_name.ext* from the same directory as the command file. When you enter *@@file_name.ext* interactively, SQL*Plus runs *file_name.ext* from the current working directory. If SQL*Plus does not find such a file, SQL*Plus searches a system-dependent path to find the file. Some operating systems may not support the path search. Consult the Oracle installation and user's manual(s) provided for your operating system for specific information related to your operating system environment.

Usage Notes

You can include in a command file any command you would normally enter interactively (typically, SQL or SQL*Plus commands).

An EXIT or QUIT command used in a command file terminates SQL*Plus.

The @@ command functions similarly to START.

If the START command is disabled, this will also disable the @@ command. For more information, see the START command later in this chapter.

SQL*Plus removes the SQLTERMINATOR (a semicolon by default) before the @@ command is issued. A workaround for this is to add another SQLTERMINATOR. See the SQLTERMINATOR variable of the SET command in this chapter for more information.

Example

Suppose that you have the following command file named PRINTRPT:

```
SELECT * FROM EMP
@EMPRPT
@@ WKRPT
```

When you START PRINTRPT and it reaches the @ command, it looks for the command file named EMPRPT in the current working directory and runs it. When PRINTRPT reaches the @@ command, it looks for the command file named WKRPT in the same path as PRINTRPT and runs it.

/ (slash)

Purpose

Executes the SQL command or PL/SQL block currently stored in the SQL buffer.

Syntax

/

Usage Notes

You can enter a slash (/) at the command prompt or at a line number prompt of a multi-line command.

The slash command functions similarly to RUN, but does not list the command in the buffer on your screen.

Executing a SQL command or PL/SQL block using the slash command will not cause the current line number in the SQL buffer to change unless the command in the buffer contains an error. In that case, SQL*Plus changes the current line number to the number of the line containing the error.

Example

To see the SQL command(s) you will execute, you can list the contents of the buffer:

```
SQL> LIST
  1 * SELECT ENAME, JOB FROM EMP WHERE ENAME = 'JAMES'
```

Enter a slash (/) at the command prompt to execute the command in the buffer:

```
SQL> /
```

For the above query, SQL*Plus displays the following output:

ENAME	JOB

JAMES	CLERK

ACCEPT

Purpose

Reads a line of input and stores it in a given user variable.

Syntax

```
ACC[EPT] variable [NUM[BER]|CHAR|DATE] [FOR[MAT] format]
[DEF[AULT] default] [PROMPT text|NOPR[OMPT]] [HIDE]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

variable

Represents the name of the variable in which you wish to store a value. If *variable* does not exist, SQL*Plus creates it.

NUM[BER]

Makes the datatype of *variable* the datatype NUMBER. If the reply does not match the datatype, ACCEPT gives an error message and prompts again.

CHAR

Makes the datatype of *variable* the datatype CHAR. The maximum CHAR length limit is 240 bytes. If a multi-byte character set is used, one CHAR may be more than one byte in size.

DATE

Makes reply a valid DATE format. If the reply is not a valid DATE format, ACCEPT gives an error message and prompts again. The datatype is CHAR.

FOR[MAT]

Specifies the input format for the reply. If the reply does not match the specified format, ACCEPT gives an error message and prompts again for a reply. The format element must be a text constant such as A10 or 9.999. See the COLUMN command in this chapter for a complete list of format elements.

Oracle date formats such as "dd/mm/yy" are valid when the datatype is DATE. DATE without a specified format defaults to the Oracle

NLS_DATE_FORMAT of the current session. See the *Oracle8i Administrator's Guide* and the *Oracle8i SQL Reference* for information on Oracle date formats.

DEF[AULT]

Sets the default value if a reply is not given. The reply must be in the specified format if defined.

PROMPT *text*

Displays *text* on-screen before accepting the value of *variable* from the user.

NOPR[OMPT]

Skips a line and waits for input without displaying a prompt.

HIDE

Suppresses the display as you type the reply.

To display or reference variables, use the DEFINE command. See the DEFINE command in this chapter for more information.

Examples

To display the prompt "Password: ", place the reply in a CHAR variable named PSWD, and suppress the display, enter

```
SQL> ACCEPT pswd CHAR PROMPT 'Password: ' HIDE
```

To display the prompt "Enter weekly salary: " and place the reply in a NUMBER variable named SALARY with a default of 000.0, enter

```
SQL> ACCEPT salary NUMBER FORMAT '999.99' DEFAULT '000.0' -  
> PROMPT 'Enter weekly salary: '
```

To display the prompt "Enter date hired: " and place the reply in a DATE variable named HIRED with the format "dd/mm/yy" and a default of "01/01/94", enter

```
SQL> ACCEPT hired DATE FORMAT 'dd/mm/yy' DEFAULT '01/01/94' -  
> PROMPT 'Enter date hired: '
```

To display the prompt "Enter employee lastname: " and place the reply in a CHAR variable named LASTNAME, enter

```
SQL> ACCEPT lastname CHAR FORMAT 'A20' -  
>          PROMPT 'Enter employee lastname:  '
```

APPEND

Purpose

Adds specified text to the end of the current line in the SQL buffer.

Syntax

```
A[PPEND] text
```

Terms and Clauses

Refer to the following for a description of the term or clause:

text

Represents the text you wish to append. If you wish to separate *text* from the preceding characters with a space, enter two spaces between APPEND and *text*.

To APPEND *text* that ends with a semicolon, end the command with two semicolons (SQL*Plus interprets a single semicolon as an optional command terminator).

Examples

To append a space and the column name DEPT to the second line of the buffer, make that line the current line by listing the line as follows:

```
SQL> 2  
      2* FROM EMP ,
```

Now enter APPEND:

```
SQL> APPEND  DEPT  
SQL> 2  
      2* FROM EMP , DEPT
```

Notice the double space between APPEND and DEPT. The first space separates APPEND from the characters to be appended; the second space becomes the first appended character.

To append a semicolon to the line, enter

```
SQL> APPEND ; ;
```

SQL*Plus appends the first semicolon to the line and interprets the second as the terminator for the APPEND command.

ARCHIVE LOG

Purpose

Starts or stops automatic archiving of online redo log files, manually (explicitly) archives specified redo log files, or displays information about redo log files.

Syntax

```
ARCHIVE LOG {LIST|STOP} [{START|NEXT|ALL|integer} [TO destination]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

LIST

Requests a display that shows the range of redo log files to be archived, the current log file group's sequence number, and the current archive destination (specified by either the optional command text or by the initialization parameter LOG_ARCHIVE_DEST).

If you are using both ARCHIVELOG mode and automatic archiving, the display might appear like:

```
SQL> ARCHIVE LOG LIST
```

Database log mode	Archive Mode
Automatic archival	Enabled
Archive destination	/vobs/oracle/dbs/arch
Oldest online log sequence	221
Next log sequence to archive	222
Current log sequence	222

Since the log sequence number of the current log group and the next log group to archive are the same, automatic archival has archived all log groups up to the current one.

If you are using ARCHIVELOG but have disabled automatic archiving, the last three lines might look like:

Oldest online log sequence	222
Next log sequence to archive	222
Current log sequence	225

If you are using NOARCHIVELOG mode, the "next log sequence to archive" line is suppressed.

The log sequence increments every time the Log Writer begins to write to another redo log file group; it does not indicate the number of logs being used. Every time an online redo log file group is reused, the contents are assigned a new log sequence number.

STOP

Disables automatic archival. If the instance is still in ARCHIVELOG mode and all redo log file groups fill, database operation is suspended until a redo log file is archived (for example, until you enter the command ARCHIVE LOG NEXT or ARCHIVE LOG ALL).

START

Enables automatic archiving. Starts the background process ARCH, which performs automatic archiving as required. If ARCH is started and a filename is supplied, the filename becomes the new default archive destination.

ARCH automatically starts on instance startup if the initialization parameter LOG_ARCHIVE_START is set to TRUE.

NEXT

Manually archives the next online redo log file group that has been filled, but not yet archived.

ALL

Manually archives all filled, but not yet archived, online log file groups.

integer

Causes archival of the online redo log file group with log sequence number *n*. You can specify any redo log file group that is still online. An error occurs if the log file cannot be found online or the sequence

number is not valid. This option can be used to rearchive a log file group.

destination

Specifies the destination device or directory in an operating system. Specification of archive destination devices is installation-specific; see your platform-specific Oracle documentation for examples of specifying archive destinations. On many operating systems, multiple log files can be spooled to the same tape.

If not specified in the command line, the archive destination is derived from the initialization parameter `LOG_ARCHIVE_DEST`. The command `ARCHIVE LOG START destination` causes the specified device or directory to become the new default archive destination for all future automatic or manual archives. A destination specified with any other option is a temporary destination that is in effect only for the current (manual) archive. It does not change the default archive destination for subsequent automatic archives.

For information about specifying archive destinations, see your platform-specific Oracle documentation.

Usage Notes

You must be connected to an open Oracle database as SYSOPER, or SYSDBA. For information about connecting to the database, see the [CONNECT](#) command.

If an online redo log file group fills and none are available for reuse, database operation is suspended. The condition can be resolved by archiving a log file group.

For information about specifying archive destinations, see your platform-specific Oracle documentation.

Note: :

This command applies only to the current instance. To specify archiving for a different instance or for all instances in a Parallel Server, use the SQL command `ALTER SYSTEM`. For more information about using SQL commands, see the [Oracle8i SQL Reference](#).

Examples

To start up the archiver process and begin automatic archiving, using the archive destination specified in LOG_ARCHIVE_DEST, enter

```
SQL> ARCHIVE LOG START
```

To stop automatic archiving, enter

```
SQL> ARCHIVE LOG STOP
```

To archive the log file group with the sequence number 1001 to the destination specified, enter

```
SQL> ARCHIVE LOG 1001 '/vobs/oracle/dbs/arch'
```

'arch' specifies the prefix of the filename on the destination device; the remainder of the filename is dependent on the initialization parameter LOG_ARCHIVE_FORMAT, which specifies the filename format for archived redo log files.

ATTRIBUTE

Purpose

Specifies display characteristics for a given attribute of an Object Type column, such as format for NUMBER data.

Also lists the current display characteristics for a single attribute or all attributes.

Syntax

```
ATTRIBUTE [type_name.attribute_name [option ...]]
```

where *option* represents one of the following clauses:

```
ALI[AS] alias  
CLE[AR]  
FOR[MAT] format  
LIKE {type_name.attribute_name | alias}  
ON | OFF
```

Terms and Clauses

Enter **ATTRIBUTE** followed by *type_name.attribute_name* and no other clauses to list the current display characteristics for only the specified attribute. Enter **ATTRIBUTE** with no clauses to list all current attribute display characteristics.

Refer to the following list for a description of each term or clause:

type_name.attribute_name

Identifies the data item (typically the name of an attribute) within the set of attributes for a given object of Object Type, *type_name*.

If you select objects of the same Object Type, an **ATTRIBUTE** command for that *type_name.attribute_name* will apply to all such objects you reference in that session.

ALI[AS] *alias*

Assigns a specified alias to a *type_name.attribute_name*, which can be used to refer to the *type_name.attribute_name* in other **ATTRIBUTE** commands.

CLE[AR]

Resets the display characteristics for the *attribute_name*. The format specification must be a text constant such as A10 or \$9,999--not a variable.

FOR[MAT] *format*

Specifies the display format of the column. The format specification must be a text constant such as A10 or \$9,999--not a variable.

LIKE {*type_name.attribute_name* | *alias*}

Copies the display characteristics of another attribute. **LIKE** copies only characteristics not defined by another clause in the current **ATTRIBUTE** command.

ON|OFF

Controls the status of display characteristics for a column. **OFF** disables the characteristics for an attribute without affecting the characteristics' definition. **ON** reinstates the characteristics.

Usage Notes

You can enter any number of **ATTRIBUTE** commands for one or more attributes. All attribute characteristics set for each attribute remain in effect for the remainder of the

session, until you turn the attribute OFF, or until you use the CLEAR COLUMN command. Thus, the ATTRIBUTE commands you enter can control an attribute's display characteristics for multiple SQL SELECT commands.

When you enter multiple ATTRIBUTE commands for the same attribute, SQL*Plus applies their clauses collectively. If several ATTRIBUTE commands apply the same clause to the same attribute, the last one entered will control the output.

Examples

To make the ENAME attribute of the Object Type EMP_TYPE 20 characters wide, enter

```
SQL> ATTRIBUTE EMP_TYPE.ENAME FORMAT A20
```

To format the SAL attribute of the Object Type EMP_TYPE so that it shows millions of dollars, rounds to cents, uses commas to separate thousands, and displays \$0.00 when a value is zero, enter

```
SQL> ATTRIBUTE EMP_TYPE.SAL FORMAT $9,999,990.99
```

BREAK

Purpose

Specifies where and how formatting will change in a report, such as

- suppressing display of duplicate values for a given column
- skipping a line each time a given column value changes
- printing COMPUTED figures each time a given column value changes or at the end of the report (see also the COMPUTE command)

Also lists the current BREAK definition.

Syntax

```
BRE[AK] [ON report_element [action [action]]] ...
```

where:

<i>report_element</i>	Requires the following syntax: <code>{column expr ROW REPORT}</code>
<i>action</i>	Requires the following syntax: <code>[SKI[P] n [SKI[P]] PAGE][<u>NODUP[LICATES]</u> DUP[LICATES]]</code>

Terms and Clauses

Refer to the following list for a description of each term or clause:

ON *column* [*action* [*action*]]

When you include action(s), specifies action(s) for SQL*Plus to take whenever a break occurs in the specified column (called the *break column*). (*column* cannot have a table or view appended to it. To achieve this, you can alias the column in the SQL statement.) A break is one of three events:

- a change in the value of a column or expression
- the output of a row
- the end of a report

When you omit action(s), BREAK ON *column* suppresses printing of duplicate values in *column* and marks a place in the report where SQL*Plus will perform the computation you specify in a corresponding COMPUTE command.

You can specify ON *column* one or more times. If you specify multiple ON clauses, as in

```
SQL> BREAK ON DEPTNO SKIP PAGE ON JOB -
> SKIP 1 ON SAL SKIP 1
```

the first ON clause represents the *outermost break* (in this case, ON DEPTNO) and the last ON clause represents the *innermost break* (in this case, ON SAL). SQL*Plus searches each row of output for the specified break(s), starting with the outermost break and proceeding--in the order

you enter the clauses--to the innermost. In the example, SQL*Plus searches for a change in the value of DEPTNO, then JOB, then SAL.

Next, SQL*Plus executes actions beginning with the action specified for the innermost break and proceeding in reverse order toward the outermost break (in this case, from SKIP 1 for ON SAL toward SKIP PAGE for ON DEPTNO). SQL*Plus executes each action up to and including the action specified for the first occurring break encountered in the initial search.

If, for example, in a given row the value of JOB changes--but the values of DEPTNO and SAL remain the same--SQL*Plus skips *two* lines before printing the row (one as a result of SKIP 1 in the ON SAL clause and one as a result of SKIP 1 in the ON JOB clause).

Whenever you use ON *column*, you should also use an ORDER BY clause in the SQL SELECT command. Typically, the columns used in the BREAK command should appear in the same order in the ORDER BY clause (although all columns specified in the ORDER BY clause need not appear in the BREAK command). This prevents breaks from occurring at meaningless points in the report.

The following SELECT command produces meaningful results:

```
SQL> SELECT DEPTNO, JOB, SAL, ENAME
      2  FROM EMP
      3  ORDER BY DEPTNO, JOB, SAL, ENAME;
```

All rows with the same DEPTNO print together on one page, and within that page all rows with the same JOB print in groups. Within each group of jobs, those jobs with the same SAL print in groups. Breaks in ENAME cause no action because ENAME does not appear in the BREAK command.

ON *expr* [*action* [*action*]]

When you include action(s), specifies action(s) for SQL*Plus to take when the value of the expression changes.

When you omit action(s), BREAK ON *expr* suppresses printing of duplicate values of *expr* and marks a place in the report where SQL*Plus will perform the computation you specify in a corresponding COMPUTE command.

You can use an expression involving one or more table columns or an alias assigned to a report column in a SQL SELECT or SQL*Plus

COLUMN command. If you use an expression in a BREAK command, you must enter *expr* exactly as it appears in the SELECT command. If the expression in the SELECT command is a+b, for example, you cannot use b+a or (a+b) in a BREAK command to refer to the expression in the SELECT command.

The information given above for ON *column* also applies to ON *expr*.

ON ROW [*action* [*action*]]

When you include action(s), specifies action(s) for SQL*Plus to take when a SQL SELECT command returns a row. The ROW break becomes the innermost break regardless of where you specify it in the BREAK command. You should always specify an action when you BREAK on a row.

ON REPORT [*action*]

Marks a place in the report where SQL*Plus will perform the computation you specify in a corresponding COMPUTE command. Use BREAK ON REPORT in conjunction with COMPUTE to print grand totals or other "grand" computed values.

The REPORT break becomes the outermost break regardless of where you specify it in the BREAK command.

Note that SQL*Plus will not skip a page at the end of a report, so you cannot use BREAK ON REPORT SKIP PAGE.

Refer to the following list for a description of each action:

SKI[P] *n*

Skips *n* lines before printing the row where the break occurred.

[SKI[P]] PAGE

Skips the number of lines that are defined to be a page before printing the row where the break occurred. The number of lines per page can be set via the PAGESIZE clause of the SET command. Note that PAGESIZE only changes the number of lines that SQL*Plus considers to be a page. Therefore, SKIP PAGE may not always cause a physical page break, unless you have also specified NEWPAGE 0. Note also that if there is a break after the last row of data to be printed in a report, SQL*Plus will not skip the page.

NODUP[LICATES]

Prints blanks rather than the value of a break column when the value is a duplicate of the column's value in the preceding row.

DUP[L ICATES]

Prints the value of a break column in every selected row.

Enter BREAK with no clauses to list the current break definition.

Usage Notes

Each new BREAK command you enter replaces the preceding one.

To remove the BREAK command, use CLEAR BREAKS.

Example

To produce a report that prints duplicate job values, prints the average of SAL and inserts one blank line when the value of JOB changes, and *additionally* prints the sum of SAL and inserts another blank line when the value of DEPTNO changes, you could enter the following commands. (The example selects departments 10 and 30 and the jobs of clerk and salesman only.)

```
SQL> BREAK ON DEPTNO SKIP 1 ON JOB SKIP 1 DUPLICATES
SQL> COMPUTE SUM OF SAL ON DEPTNO
SQL> COMPUTE AVG OF SAL ON JOB
SQL> SELECT DEPTNO, JOB, ENAME, SAL FROM EMP
      2 WHERE JOB IN ('CLERK', 'SALESMAN')
      3 AND DEPTNO IN (10, 30)
      4 ORDER BY DEPTNO, JOB;
```

The following output results:

DEPTNO	JOB	ENAME	SAL
10	CLERK	MILLER	1300

	avg		1300

	sum		1300
30	CLERK	JAMES	1045

	avg		1045
	SALESMAN	ALLEN	1760
	SALESMAN	MARTIN	1375
	SALESMAN	TURNER	1650

SALESMAN	WARD	1375
*****		-----
avg		1540
*****		-----
sum		7205

BTITLE

Purpose

Places and formats a specified title at the bottom of each report page or lists the current BTITLE definition.

For a description of the old form of BTITLE, see Appendix F.

Syntax

BTI[TLE] [*printspec* [*text*|*variable*] ...] | [OFF|ON]

Terms and Clauses

Refer to the TTITLE command for additional information on terms and clauses in the BTITLE command syntax.

Enter BTITLE with no clauses to list the current BTITLE definition.

Usage Notes

If you do not enter a *printspec* clause before the first occurrence of *text*, BTITLE left justifies the text. SQL*Plus interprets BTITLE in the new form if a valid *printspec* clause (LEFT, SKIP, COL, and so on) immediately follows the command name.

Examples

To set a bottom title with CORPORATE PLANNING DEPARTMENT on the left and a date on the right, enter

```
SQL> BTITLE LEFT 'CORPORATE PLANNING DEPARTMENT' -
> RIGHT '23 Nov 1998'
```

To set a bottom title with CONFIDENTIAL in column 50, followed by six spaces and a date, enter

```
SQL> BTITLE COL 50 'CONFIDENTIAL' TAB 6 '23 Nov 1998'
```

CHANGE

Purpose

Changes the first occurrence of text on the current line in the buffer.

Syntax

```
C[CHANGE] sepchar old [sepchar [new [sepchar]]]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

sepchar

Represents any non-alphanumeric character such as "/" or "!". Use a *sepchar* that does not appear in *old* or *new*. You can omit the space between CHANGE and the first *sepchar*.

old

Represents the text you wish to change. CHANGE ignores case in searching for *old*. For example,

```
CHANGE /aq/aw
```

will find the first occurrence of "aq", "AQ", "aQ", or "Aq" and change it to "aw". SQL*Plus inserts the *new* text exactly as you specify it.

If *old* is prefixed with "...", it matches everything up to and including the first occurrence of *old*. If it is suffixed with "...", it matches the first occurrence of *old* and everything that follows on that line. If it contains an embedded "...", it matches everything from the preceding part of *old* through the following part of *old*.

new

Represents the text with which you wish to replace *old*. If you omit *new* and, optionally, the second and third *sepchars*, CHANGE deletes *old* from the current line of the buffer.

Usage Notes

CHANGE changes the first occurrence of the existing specified text on the current line of the buffer to the new specified text. The current line is marked with an asterisk (*) in the LIST output.

You can also use CHANGE to modify a line in the buffer that has generated an Oracle error. SQL*Plus sets the buffer's current line to the line containing the error so that you can make modifications.

To reenter an entire line, you can type the line number followed by the new contents of the line. If you specify a line number larger than the number of lines in the buffer and follow the number with text, SQL*Plus adds the text in a new line at the end of the buffer. If you specify zero ("0") for the line number and follow the zero with text, SQL*Plus inserts the line at the beginning of the buffer (that line becomes line 1).

Examples

Assume the current line of the buffer contains the following text:

```
4* WHERE JOB IS IN ( 'CLERK' , 'SECRETARY' , 'RECEPTIONIST' )
```

Enter the following command:

```
SQL> C /RECEPTIONIST/GUARD/
```

The text in the buffer changes as follows:

```
4* WHERE JOB IS IN ( 'CLERK' , 'SECRETARY' , 'GUARD' )
```

Or enter the following command:

```
SQL> C / 'CLERK' , ... / 'CLERK' ) /
```

The original line changes to

```
4* WHERE JOB IS IN ( 'CLERK' )
```

Or enter the following command:

```
SQL> C / ( ... ) / ( 'COOK' , 'BUTLER' ) /
```

The original line changes to

```
4* WHERE JOB IS IN ( 'COOK' , 'BUTLER' )
```

You can replace the contents of an entire line using the line number. This entry

```
SQL> 2 FROM EMP e1
```

causes the second line of the buffer to be replaced with

```
FROM EMP e1
```

Note that entering a line number followed by a string will replace the line regardless of what text follows the line number. For example,

```
SQL> 2 c/old/new/
```

will change the second line of the buffer to be

```
2* c/old/new/
```

CLEAR

Purpose

Resets or erases the current value or setting for the specified option.

Syntax

```
CL[EAR] option ...
```

where *option* represents one of the following clauses:

```
BRE[AKS]  
BUFF[ER]  
COL[UMNS]  
COMP[UTES]  
SCR[EEN]  
SQL  
TIMI[NG]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

BRE[AKS]

Removes the break definition set by the BREAK command.

BUFF[ER]

Clears text from the buffer. CLEAR BUFFER has the same effect as CLEAR SQL, unless you are using multiple buffers (see the SET BUFFER command in Appendix F).

COL[UMNS]

Resets column display attributes set by the COLUMN command to default settings for all columns. To reset display attributes for a single column, use the CLEAR clause of the COLUMN command. CLEAR COLUMNS also clears the ATTRIBUTES for that column.

COMP[UTES]

Removes all COMPUTE definitions set by the COMPUTE command.

SCR[EEN]

Clears your screen.

SQL

Clears the text from SQL buffer. CLEAR SQL has the same effect as CLEAR BUFFER, unless you are using multiple buffers (see the SET BUFFER command in Appendix F).

TIMI[NG]

Deletes all timers created by the TIMING command.

Example

To clear breaks, enter

```
SQL> CLEAR BREAKS
```

To clear column definitions, enter

```
SQL> CLEAR COLUMNS
```

COLUMN**Purpose**

Specifies display attributes for a given column, such as

- text for the column heading
- alignment of the column heading
- format for NUMBER data
- wrapping of column data

Also lists the current display attributes for a single column or all columns.

Syntax

```
COL[UMN] [{column|expr} [option ...]]
```

where *option* represents one of the following clauses:

```
ALI[AS] alias
CLE[AR]
FOLD_A[FTER]
FOLD_B[EFORE]
FOR[MAT] format
HEA[DING] text
JUS[TIFY] {L[EFT]|C[ENTER]|C[ENTRE]|R[IGHT]}
LIKE {expr|alias}
NEWL[INE]
NEW_V[ALUE] variable
NOPRI[NT]|PRI[NT]
NUL[L] text
OLD_V[ALUE] variable
ON|OFF
WRA[PPED]|WOR[D_WRAPPED]|TRU[NCATED]
```

Terms and Clauses

Enter COLUMN followed by *column* or *expr* and no other clauses to list the current display attributes for only the specified column or expression. Enter COLUMN with no clauses to list all current column display attributes.

Refer to the following list for a description of each term or clause:

{*column*|*expr*}

Identifies the data item (typically, the name of a column) in a SQL SELECT command to which the column command refers. If you use an expression in a COLUMN command, you must enter *expr* exactly as it appears in the SELECT command. If the expression in the SELECT command is a+b, for example, you cannot use b+a or (a+b) in a COLUMN command to refer to the expression in the SELECT command.

If you select columns with the same name from different tables, a COLUMN command for that column name will apply to both columns. That is, a COLUMN command for the column ENAME applies to all columns named ENAME that you reference in this session. COLUMN ignores table name prefixes in SELECT commands. Also, spaces are ignored unless the name is placed in double quotes.

To format the columns differently, assign a unique alias to each column within the SELECT command itself (do not use the ALIAS clause of the COLUMN command) and enter a COLUMN command for each column's alias.

ALI[AS] *alias*

Assigns a specified *alias* to a column, which can be used to refer to the column in BREAK, COMPUTE, and other COLUMN commands.

CLE[AR]

Resets the display attributes for the column to default values.

To reset the attributes for all columns, use the CLEAR COLUMNS command. CLEAR COLUMNS also clears the ATTRIBUTES for that column.

FOLD_A[FTER]

Inserts a carriage return after the column heading and after each row in the column. SQL*Plus does not insert an extra carriage return after the last column in the SELECT list.

FOLD_B[EFORE]

Inserts a carriage return before the column heading and before each row of the column. SQL*Plus does not insert an extra carriage return before the first column in the SELECT list.

FOR[MAT] *format*

Specifies the display format of the column. The format specification must be a text constant such as A10 or \$9,999--not a variable.

Character Columns The default width of CHAR, NCHAR, VARCHAR2 (VARCHAR) and NVARCHAR2 (NCHAR VARYING) columns is the width of the column in the database. SQL*Plus formats these datatypes left-justified. If a value does not fit within the column width, SQL*Plus wraps or truncates the character string depending on the setting of SET WRAP.

A LONG, CLOB or NCLOB column's width defaults to the value of SET LONGCHUNKSIZE or SET LONG, whichever one is smaller.

To change the width of a datatype to *n*, use FORMAT *An*. (A stands for alphanumeric.) If you specify a width shorter than the column heading, SQL*Plus truncates the heading. If you specify a width for a LONG, CLOB, or NCLOB column, SQL*Plus uses the LONGCHUNKSIZE or the specified width, whichever is smaller, as the column width.

DATE Columns The default width and format of unformatted DATE columns in SQL*Plus is derived from the NLS parameters in effect. Otherwise, the default width is A9. In Oracle8, the NLS parameters may be set in your database parameter file or may be environment variables or an equivalent platform-specific mechanism. They

may also be specified for each session with the ALTER SESSION command. (See the documentation for Oracle8i for a complete description of the NLS parameters).

You can change the format of any DATE column using the SQL function TO_CHAR in your SQL SELECT statement. You may also wish to use an explicit COLUMN FORMAT command to adjust the column width.

When you use SQL functions like TO_CHAR, Oracle automatically allows for a very wide column.

To change the width of a DATE column to *n*, use the COLUMN command with FORMAT An. If you specify a width shorter than the column heading, the heading is truncated.

NUMBER Columns To change a NUMBER column's width, use FORMAT followed by an element as specified in [Table 8-1](#).

Table 8-1 Number Formats

Element	Example(s)	Description
9	9999	Number of "9"s specifies number of significant digits returned. Blanks are displayed for leading zeroes. A zero (0) is displayed for a value of zero.
0	0999 9990	Displays a leading zero or a value of zero in this position as 0.
\$	\$9999	Prefixes value with dollar sign.
B	B9999	Displays a zero value as blank, regardless of "0"s in the format model.
MI	9999MI	Displays "-" after a negative value. For a positive value, a trailing space is displayed.
S	S9999	Returns "+" for positive values and "-" for negative values in this position.
PR	9999PR	Displays a negative value in <angle brackets>. For a positive value, a leading and trailing space is displayed.
D	99D99	Displays the decimal character in this position, separating the integral and fractional parts of a number.
G	9G999	Displays the group separator in this position.
C	C999	Displays the ISO currency symbol in this position.
L	L999	Displays the local currency symbol in this position.
, (comma)	9,999	Displays a comma in this position.

.	99.99	Displays a period (decimal point) in this position, separating the integral and fractional parts of a number.
V	999V99	Multiplies value by 10 ⁿ , where <i>n</i> is the number of "9"s after the "V".
EEEE	9.999EEEE	Displays value in scientific notation (format must contain exactly four "E"s).
RN or rn	RN	Displays upper- or lowercase Roman numerals. Value can be an integer between 1 and 3999.
DATE	DATE	Displays value as a date in MM/DD/YY format; used to format NUMBER columns that represent Julian dates.

The MI and PR format elements can only appear in the last position of a number format model. The S format element can only appear in the first or last position.

If a number format model does not contain the MI, S or PR format elements, negative return values automatically contain a leading negative sign and positive values automatically contain a leading space.

A number format model can contain only a single decimal character (D) or period (.), but it can contain multiple group separators (G) or commas (,). A group separator or comma cannot appear to the right of a decimal character or period in a number format model.

SQL*Plus formats NUMBER data right-justified. A NUMBER column's width equals the width of the heading or the width of the FORMAT plus one space for the sign, whichever is greater. If you do not explicitly use FORMAT, then the column's width will always be at least the value of SET NUMWIDTH.

SQL*Plus may round your NUMBER data to fit your format or field width.

If a value cannot fit within the column width, SQL*Plus indicates overflow by displaying a pound sign (#) in place of each digit the width allows.

If a positive value is extremely large and a numeric overflow occurs when rounding a number, then the infinity sign (~) replaces the value. Likewise, if a negative value is extremely small and a numeric overflow occurs when rounding a number, then the negative infinity sign replaces the value (-~).

HEA[DING] text

Defines a column heading. If you do not use a HEADING clause, the column's heading defaults to *column* or *expr*. If *text* contains blanks or punctuation characters, you must enclose it with single or double quotes. Each occurrence of the HEADSEP character (by default, "|") begins a new line.

For example,

COLUMN ENAME HEADING 'Employee |Name'

would produce a two-line column heading. See the HEADSEP variable of the SET command in this chapter for information on changing the HEADSEP character.

JUS[TIFY] {L[EFT] | C[ENTER] | C[ENTRE] | R[IGHT]}

Aligns the heading. If you do not use a JUSTIFY clause, headings for NUMBER columns default to RIGHT and headings for other column types default to LEFT.

LIKE {*expr* | *alias*}

Copies the display attributes of another column or expression (whose attributes you have already defined with another COLUMN command). LIKE copies only attributes not defined by another clause in the current COLUMN command.

NEWL[INE]

Starts a new line before displaying the column's value. NEWLINE has the same effect as FOLD_BEFORE.

NEW_V[ALUE] *variable*

Specifies a variable to hold a column value. You can reference the variable in TTITLE commands. Use NEW_VALUE to display column values or the date in the top title. You must include the column in a BREAK command with the SKIP PAGE action. The variable name cannot contain a pound sign (#).

NEW_VALUE is useful for master/detail reports in which there is a new master record for each page. For master/detail reporting, you must also include the column in the ORDER BY clause. See the example at the end of this command description.

For information on displaying a column value in the bottom title, see COLUMN OLD_VALUE. For more information on referencing variables in titles, see the TTITLE command later in this chapter. For information on formatting and valid format models, see the COLUMN FORMAT command.

NOPRI[NT] | PRI[NT]

Controls the printing of the column (the column heading and all the selected values). NOPRINT turns the printing of the column off. PRINT turns the printing of the column on.

NUL[L] *text*

Controls the text SQL*Plus displays for null values in the given column. The default is a white space. SET NULL controls the text displayed for all null values for all columns, unless overridden for a specific column by the NULL clause of the COLUMN command. When a NULL value is SELECTed, a variable's type will always become CHAR so the SET NULL text can be stored in it.

OLD_V[ALUE] *variable*

Specifies a variable to hold a column value. You can reference the variable in BTITLE commands. Use OLD_VALUE to display column values in the bottom title. You must include the column in a BREAK command with the SKIP PAGE action.

OLD_VALUE is useful for master/detail reports in which there is a new master record for each page. For master/detail reporting, you must also include the column in the ORDER BY clause.

For information on displaying a column value in the top title, see COLUMN NEW_VALUE. For more information on referencing variables in titles, see the TTITLE command later in this chapter.

ON|OFF

Controls the status of display attributes for a column. OFF disables the attributes for a column without affecting the attributes' definition. ON reinstates the attributes.

WRA[PPED] | WOR[D_WRAPPED] | TRU[NCATED]

Specifies how SQL*Plus will treat a datatype or DATE string that is too wide for a column. WRAPPED wraps the string within the column bounds, beginning new lines when required. When WORD_WRAP is enabled, SQL*Plus left justifies each new line, skipping all leading whitespace (for example, returns, newline characters, tabs and spaces), including embedded newline characters. Embedded whitespace not on a line boundary is not skipped. TRUNCATED truncates the string at the end of the first line of display.

Usage Notes

You can enter any number of COLUMN commands for one or more columns. All column attributes set for each column remain in effect for the remainder of the session, until you turn the column OFF, or until you use the CLEAR COLUMN command. Thus, the COLUMN commands you enter can control a column's display attributes for multiple SQL SELECT commands.

When you enter multiple COLUMN commands for the same column, SQL*Plus applies their clauses collectively. If several COLUMN commands apply the same clause to the same column, the last one entered will control the output.

Examples

To make the ENAME column 20 characters wide and display EMPLOYEE NAME on two lines at the top, enter

```
SQL> COLUMN ENAME FORMAT A20 HEADING 'EMPLOYEE |NAME'
```

To format the SAL column so that it shows millions of dollars, rounds to cents, uses commas to separate thousands, and displays \$0.00 when a value is zero, you would enter

```
SQL> COLUMN SAL FORMAT $9,999,990.99
```

To assign the alias NET to a column containing a long expression, to display the result in a dollar format, and to display <NULL> for null values, you might enter

```
SQL> COLUMN SAL+COMM+BONUS-EXPENSES-INS-TAX ALIAS NET
SQL> COLUMN NET FORMAT $9,999,999.99 NULL '<NULL>'
```

Note that the example divides this column specification into two commands. The first defines the alias NET, and the second uses NET to define the format.

Also note that in the first command you must enter the expression exactly as you entered it (or will enter it) in the SELECT command. Otherwise, SQL*Plus cannot match the COLUMN command to the appropriate column.

To wrap long values in a column named REMARKS, you can enter

```
SQL> COLUMN REMARKS FORMAT A20 WRAP
```

For example:

CUSTOMER	DATE	QUANTITY	REMARKS
123	25-AUG-86	144	This order must be shipped by air freight to ORD

If you replace WRAP with WORD_WRAP, REMARKS looks like this:

CUSTOMER	DATE	QUANTITY	REMARKS
123	25-AUG-86	144	This order must be shipped by air freight to ORD

If you specify TRUNCATE, REMARKS looks like this:

CUSTOMER	DATE	QUANTITY	REMARKS
123	25-AUG-86	144	This order must be s

In order to print the current date and the name of each job in the top title, enter the following. (For details on creating a date variable, see ["Displaying the Current Date in Titles"](#) under ["Defining Page and Report Titles and Dimensions"](#) in [Chapter 4](#).)

```
SQL> COLUMN JOB NOPRINT NEW_VALUE JOBVAR
SQL> COLUMN TODAY NOPRINT NEW_VALUE DATEVAR
SQL> BREAK ON JOB SKIP PAGE ON TODAY
SQL> TTITLE CENTER 'Job Report' RIGHT DATEVAR SKIP 2 -
> LEFT 'Job: ' JOBVAR SKIP 2
SQL> SELECT TO_CHAR(SYSDATE, 'MM/DD/YY') TODAY,
2 ENAME, JOB, MGR, HIREDATE, SAL, DEPTNO
3 FROM EMP WHERE JOB IN ('CLERK', 'SALESMAN')
4 ORDER BY JOB, ENAME;
```

Your two page report would look similar to the following report, with "Job Report" centered within your current linesize:

```

                                Job Report                                11/23/98
Job:      CLERK

ENAME      MGR  HIREDATE      SAL      DEPTNO
-----
ADAMS      7788 14-JAN-87      1100      20
JAMES      7698 03-DEC-81       950      30
MILLER     7782 23-JAN-82      1300      10
SMITH      7902 17-DEC-80       800      20

```

```

                                Job Report                                11/23/98
Job:      CLERK

ENAME      MGR  HIREDATE      SAL      DEPTNO
-----
ALLEN      7698 20-JAN-81      1600      30
MARTIN     7698 03-DEC-81       950      30
MILLER     7782 23-JAN-82      1300      10
SMITH      7902 17-DEC-80       800      20

```

To change the default format of DATE columns to 'YYYY-MM-DD', you can enter

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY-MM-DD' ;
```

The following output results:

```
Session altered
```

To display the change, enter a SELECT statement, such as:

```
SQL> SELECT HIREDATE
2  FROM EMP
3  WHERE EMPNO = 7839 ;
```

The following output results:

```
HIREDATE
-----
1981-11-17
```

See the *Oracle8i SQL Reference* for information on the ALTER SESSION command.

Note that in a SELECT statement, some SQL calculations or functions, such as TO_CHAR, may cause a column to be very wide. In such cases, use the FORMAT option to alter the column width.

COMPUTE

Purpose

Calculates and prints summary lines, using various standard computations, on subsets of selected rows, or lists all COMPUTE definitions. (For details on how to create summaries, see "[Clarifying Your Report with Spacing and Summary Lines](#)" in [Chapter 4](#).)

Syntax

```
COMP[UTE] [function [LAB[EL] text] ...
  OF {expr|column|alias} ...
  ON {expr|column|alias|REPORT|ROW} ...]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

***function* ...**

Represents one of the functions listed in [Table 8-2](#). If you specify more than one function, use spaces to separate the functions.

Table 8-2 COMPUTE Functions

Function	Computes	Applies to Datatypes
AVG	Average of non-null values	NUMBER
COU[NT]	Count of non-null values	all types
MAX[IMUM]	Maximum value	NUMBER, CHAR, NCHAR, VARCHAR2 (VARCHAR), NVARCHAR2 (NCHAR VARYING)
MIN[IMUM]	Minimum value	NUMBER, CHAR, NCHAR, VARCHAR2 (VARCHAR), NVARCHAR2 (NCHAR VARYING)
NUM[BER]	Count of rows	all types
STD	Standard deviation of non-null values	NUMBER
SUM	Sum of non-null values	NUMBER
VAR[IANCE]	Variance of non-null values	NUMBER

LAB[EL] *text*

Defines the label to be printed for the computed value. If no LABEL clause is used, *text* defaults to the unabbreviated function keyword. If *text* contains spaces or punctuation, you must enclose it with single quotes. The label

prints left justified and truncates to the column width or linesize, whichever is smaller. The maximum length of a label is 500 characters.

The label for the computed value appears in the break column specified. To suppress the label, use the NOPRINT option of the COLUMN command on the break column.

If you repeat a function in a COMPUTE command, SQL*Plus issues a warning and uses the first occurrence of the function.

With ON REPORT and ON ROW computations, the label appears in the first column listed in the SELECT statement. The label can be suppressed by using a NOPRINT column first in the SELECT statement. When you compute a function of the first column in the SELECT statement ON REPORT or ON ROW, then the computed value appears in the first column and the label is not displayed. To see the label, select a dummy column first in the SELECT list.

OF {*expr*|*column*|*alias*} ...

In the OF clause, you can refer to an expression or function reference in the SELECT statement by placing the expression or function reference in double quotes. Column names and aliases do not need quotes.

ON {*expr*|*column*|*alias*|**REPORT**|**ROW**} ...

Specifies the event SQL*Plus will use as a break. (*column* cannot have a table or view appended to it. To achieve this, you can alias the column in the SQL statement.) COMPUTE prints the computed value and restarts the computation when the event occurs (that is, when the value of the expression changes, a new ROW is fetched, or the end of the report is reached).

If multiple COMPUTE commands reference the same column in the ON clause, only the last COMPUTE command applies.

To reference a SQL SELECT expression or function reference in an ON clause, place the expression or function reference in quotes. Column names and aliases do not need quotes.

Enter COMPUTE without clauses to list all COMPUTE definitions.

Usage Notes

In order for the computations to occur, the following conditions must all be true:

- One or more of the expressions, columns, or column aliases you reference in the OF clause must also be in the SELECT command.
- The expression, column, or column alias you reference in the ON clause must occur in the SELECT command and in the most recent BREAK command.
- If you reference either ROW or REPORT in the ON clause, also reference ROW or REPORT in the most recent BREAK command.

To remove all COMPUTE definitions, use the CLEAR COMPUTES command.

Examples

To subtotal the salary for the "clerk", "analyst", and "salesman" job classifications with a compute label of "TOTAL", enter

```
SQL> BREAK ON JOB SKIP 1
SQL> COMPUTE SUM LABEL 'TOTAL' OF SAL ON JOB
SQL> SELECT JOB, ENAME, SAL
      2 FROM EMP
      3 WHERE JOB IN ('CLERK', 'ANALYST', 'SALESMAN')
      4 ORDER BY JOB, SAL;
```

The following output results:

JOB	ENAME	SAL
ANALYST	SCOTT	3000
	FORD	3000
*****		-----
TOTAL		6000
CLERK	SMITH	800
	JAMES	950
	ADAMS	1100
	MILLER	1300
*****		-----
TOTAL		4150
SALESMAN	WARD	1250
	MARTIN	1250
	TURNER	1500
	ALLEN	1600
*****		-----
TOTAL		5600

To calculate the total of salaries less than 1,000 on a report, enter

```
SQL> COMPUTE SUM OF SAL ON REPORT
SQL> BREAK ON REPORT
SQL> COLUMN DUMMY HEADING ''
SQL> SELECT ' ' DUMMY, SAL, EMPNO
      2 FROM EMP
      3 WHERE SAL < 1000
      4 ORDER BY SAL;
```

The following output results:

	SAL	EMPNO
	800	7369
	950	7900

sum	5350	

To compute the average and maximum salary for the accounting and sales departments, enter

```
SQL> BREAK ON DNAME SKIP 1
SQL> COMPUTE AVG LABEL 'Dept Average' -
>          MAX LABEL 'Dept Maximum' -
>          OF SAL ON DNAME
SQL> SELECT DNAME, ENAME, SAL
2  FROM DEPT, EMP
3  WHERE DEPT.DEPTNO = EMP.DEPTNO
4  AND DNAME IN ('ACCOUNTING', 'SALES')
5  ORDER BY DNAME;
```

The following output results:

DNAME	ENAME	SAL

ACCOUNTING	CLARK	2450
	KING	5000
	MILLER	1300

Dept Average		2916.66667
Dept Maximum		5000
SALES	ALLEN	1600
	WARD	1250
	JAMES	950
	TURNER	1500
	MARTIN	1250
	BLAKE	2850

Dept Average		1566.66667
Dept Maximum		2850

To compute the sum of salaries for departments 10 and 20 without printing the compute label:

```
SQL> COLUMN DUMMY NOPRINT
SQL> COMPUTE SUM OF SAL ON DUMMY
SQL> BREAK ON DUMMY SKIP 1
SQL> SELECT DEPTNO DUMMY, DEPTNO, ENAME, SAL
2  FROM EMP
3  WHERE DEPTNO <= 20
4  ORDER BY DEPTNO;
```

SQL*Plus displays the following output:

DEPTNO	ENAME	SAL

10	KING	5000
10	CLARK	2450
10	MILLER	1300

		8750
20	JONES	2975
20	FORD	3000
20	SMITH	800
20	SCOTT	3000
20	ADAMS	1100

		10875

If, instead, you do not want to print the label, only the salary total at the end of the report:

```
SQL> COLUMN DUMMY NOPRINT
SQL> COMPUTE SUM OF SAL ON DUMMY
SQL> BREAK ON DUMMY
SQL> SELECT NULL DUMMY, DEPTNO, ENAME, SAL
  2 FROM EMP
  3 WHERE DEPTNO <= 20
  4 ORDER BY DEPTNO;
```

SQL*Plus displays the following output:

DEPTNO	ENAME	SAL
10	KING	5000
10	CLARK	2450
10	MILLER	1300
20	JONES	2975
20	FORD	3000
20	SMITH	800
20	SCOTT	3000
20	ADAMS	1100
		19625

CONNECT

Purpose

Connects a given username to Oracle.

Syntax

```
CONN[ECT]  [[logon] [AS [SYSOPER|SYSDBA]]]
```

where *logon* requires the following syntax:

```
username[/password][@net_service_name]/
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

***username*[/*password*]**

Represent the username and password with which you wish to connect to Oracle. If you omit *username* and *password*, SQL*Plus prompts you for them. If you enter a slash (/) or simply enter [Return] to the prompt for *username*, SQL*Plus logs you in using a default logon (see "/" below).

If you omit only *password*, SQL*Plus prompts you for *password*. When prompting, SQL*Plus does not display *password* on your terminal screen. See the PASSWORD command in this chapter for information about changing your password.

net_service_name

Consists of a Net8 connection string. The exact syntax depends upon the Net8 communications protocol your Oracle installation uses. For more information, refer to the Net8 manual appropriate for your protocol or contact your DBA. SQL*Plus does not prompt for a service name, but uses your default database if you do not include a specification.

/

Represents a default logon using operating system authentication. You cannot enter a *net_service_name* if you use a default logon. In a default logon, SQL*Plus typically attempts to log you in using the username OPS\$name, where *name* is your operating system username. See the *Oracle8i Administrator's Guide* for information about operating system authentication.

AS [SYSOPER|SYSDBA]

The AS clause allows privileged connections by users who have been granted SYSOPER or SYSDBA system privileges.

Usage Notes

CONNECT commits the current transaction to the database, disconnects the current username from Oracle, and reconnects with the specified username.

If you log on or connect as a user whose account has expired, SQL*Plus prompts you to change your password before you can connect.

If an account is locked, a message is displayed and connection into that account (as that user) is not permitted until the account is unlocked by your DBA.

For more information about user account management, refer to the CREATE and ALTER USER commands, and the CREATE PROFILE command in the *Oracle8i SQL Reference*.

Examples

To connect across Net8 using username SCOTT and password TIGER to the database known by the Net8 alias as FLEETDB, enter

```
SQL> CONNECT SCOTT/TIGER@FLEETDB
```

To connect using username SCOTT, letting SQL*Plus prompt you for the password, enter

```
SQL> CONNECT SCOTT
```

For more information about setting up your password file, refer to the *Oracle8i Administrator's Guide*.

To use a password file to connect to an instance on the current node as a privileged user named SCOTT with the password TIGER, enter

```
SQL> CONNECT SCOTT/TIGER AS SYSDBA
```

Note, that your default schema is now SYS, not SCOTT.

COPY

Purpose

Copies the data from a query to a table in a local or remote database.

Syntax

```
COPY {FROM username[/password]@net_service_name |  
      TO username[/password]@net_service_name |  
      FROM username[/password]@net_service_name  
      TO username[/password]@net_service_name}  
{APPEND|CREATE|INSERT|REPLACE} destination_table  
[(column, column, column ...)] USING query
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

username[/password]

Represent the Oracle *username/password* you wish to COPY FROM and TO. In the FROM clause, *username/password* identifies the source of the data; in the TO clause, *username/password* identifies the destination. If you do not specify *password* in either the FROM clause or the TO clause, SQL*Plus will prompt you for it. SQL*Plus suppresses the display of your response to these prompts.

net_service_name

Consists of a Net8 connection string. You must include a *net_service_name* clause in the COPY command. In the FROM clause, *net_service_name* represents the database at the source; in the TO clause, *net_service_name* represents the database at the destination. The exact syntax depends upon the Net8 communications protocol your Oracle installation uses. For more information, refer to the Net8 manual appropriate for your protocol or contact your DBA.

destination_table

Represents the table you wish to create or to which you wish to add data.

(column, column, column, ...)

Specifies the names of the columns in *destination_table*. You must enclose a name in double quotes if it contains lowercase letters or blanks.

If you specify columns, the number of columns must equal the number of columns selected by the query. If you do not specify any columns, the copied columns will

have the same names in the destination table as they had in the source if COPY creates *destination_table*.

USING query

Specifies a SQL query (SELECT command) determining which rows and columns COPY copies.

FROM username [/password] @net_service_name

Specifies the username, password, and database that contains the data to be copied. If you omit the FROM clause, the source defaults to the database to which SQL*Plus is connected (that is, the database that other commands address). You must include a FROM clause to specify a source database other than the default.

TO username [/password] @net_service_name

Specifies the database containing the destination table. If you omit the TO clause, the destination defaults to the database to which SQL*Plus is connected (that is, the database that other commands address). You must include a TO clause to specify a destination database other than the default.

APPEND

Inserts the rows from *query* into *destination_table* if the table exists. If *destination_table* does not exist, COPY creates it.

CREATE

Inserts the rows from *query* into *destination_table* after first creating the table. If *destination_table* already exists, COPY returns an error.

INSERT

Inserts the rows from *query* into *destination_table*. If *destination_table* does not exist, COPY returns an error. When using INSERT, the USING *query* must select one column for each column in the *destination_table*.

REPLACE

Replaces *destination_table* and its contents with the rows from *query*. If *destination_table* does not exist, COPY creates it. Otherwise, COPY drops the existing table and replaces it with a table containing the copied data.

Usage Notes

To enable the copying of data between Oracle and non-Oracle databases, NUMBER columns are changed to DECIMAL columns in the destination table. Hence, if you are copying between Oracle databases, a NUMBER column with no precision will be changed to a DECIMAL(38) column. When copying between Oracle databases, you should use SQL commands (CREATE TABLE AS and INSERT) or you should ensure that your columns have a precision specified.

The SQL*Plus SET variable LONG limits the length of LONG columns that you copy. If any LONG columns contain data longer than the value of LONG, COPY truncates the data.

SQL*Plus performs a commit at the end of each successful COPY. If you set the SQL*Plus SET variable COPYCOMMIT to a positive value *n*, SQL*Plus performs a commit after copying every *n* batches of records. The SQL*Plus SET variable ARRAYSIZE determines the size of a batch.

Some operating environments require that service names be placed in double quotes.

Examples

The following command copies the entire EMP table to a table named WESTEMP. Note that the tables are located in two different databases. If WESTEMP already exists, SQL*Plus replaces the table and its contents. The columns in WESTEMP have the same names as the columns in the source table, EMP.

```
SQL> COPY FROM SCOTT/TIGER@HQ TO JOHN/CHROME@WEST -  
> REPLACE WESTEMP -  
> USING SELECT * FROM EMP
```

The following command copies selected records from EMP to the database to which SQL*Plus is connected. SQL*Plus creates SALESMEN through the copy. SQL*Plus copies only the columns EMPNO and ENAME, and at the destination names them EMPNO and SALESMAN.

```
SQL> COPY FROM SCOTT/TIGER@HQ -  
> CREATE SALESMEN (EMPNO, SALESMAN) -  
> USING SELECT EMPNO, ENAME FROM EMP -  
> WHERE JOB= 'SALESMAN'
```

DEFINE

Purpose

Specifies a user variable and assigns it a CHAR value, or lists the value and variable type of a single variable or all variables.

Syntax

```
DEF[INE] [variable] | [variable = text]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

variable

Represents the user variable whose value you wish to assign or list.

text

Represents the CHAR value you wish to assign to *variable*. Enclose *text* in single quotes if it contains punctuation or blanks.

```
variable = text
```

Defines (names) a user variable and assigns it a CHAR value.

Enter DEFINE followed by *variable* to list the value and type of *variable*. Enter DEFINE with no clauses to list the values and types of all user variables.

Usage Notes

DEFINED variables retain their values until one of the following events occurs:

- you enter a new DEFINE command referencing the variable
- you enter an UNDEFINE command referencing the variable
- you enter an ACCEPT command referencing the variable
- you reference the variable in the NEW_VALUE or OLD_VALUE clause of the COLUMN command and reference the column in a subsequent SQL SELECT command
- you EXIT SQL*Plus

Whenever you run a stored query or command file, SQL*Plus substitutes the value of *variable* for each substitution variable referencing *variable* (in the form *&variable* or *&&variable*). SQL*Plus will not prompt you for the value of *variable* in this session until you UNDEFINE *variable*.

Note that you can use DEFINE to define the variable, _EDITOR, which establishes the host system editor invoked by the SQL*Plus EDIT command.

If you continue the value of a DEFINED variable on multiple lines (using the SQL*Plus command continuation character), SQL*Plus replaces each continuation character and carriage return you enter with a space in the resulting variable. For example, SQL*Plus interprets

```
SQL> DEFINE TEXT = 'ONE-  
> TWO-  
> THREE'
```

as

```
SQL> DEFINE TEXT = 'ONE TWO THREE'
```

Examples

To assign the value MANAGER to the variable POS, type:

```
SQL> DEFINE POS = MANAGER
```

If you execute a command that contains a reference to &POS, SQL*Plus will substitute the value MANAGER for &POS and will not prompt you for a POS value.

To assign the CHAR value 20 to the variable DEPTNO, type:

```
SQL> DEFINE DEPTNO = 20
```

Even though you enter the number 20, SQL*Plus assigns a CHAR value to DEPTNO consisting of two characters, 2 and 0.

To list the definition of DEPTNO, enter

```
SQL> DEFINE DEPTNO
DEFINE DEPTNO          = "20" (CHAR)
```

This result shows that the value of DEPTNO is 20.

DEL

Purpose

Deletes one or more lines of the buffer.

Syntax

```
DEL [n|n m|n *|n LAST|*|* n|* LAST|LAST]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

n	Deletes line <i>n</i> .
n m	Deletes lines <i>n</i> through <i>m</i> .
n *	Deletes line <i>n</i> through the current line.
n LAST	Deletes line <i>n</i> through the last line.
*	Deletes the current line.
* n	Deletes the current line through line <i>n</i> .
* LAST	Deletes the current line through the last line.
LAST	Deletes the last line.

Enter DEL with no clauses to delete the current line of the buffer.

Usage Notes

DEL makes the following line of the buffer (if any) the current line. You can enter DEL several times to delete several consecutive lines.

Note:

DEL is a SQL*Plus command and DELETE is a SQL command. For more information about the SQL DELETE command, see the [Oracle8i SQL Reference](#).

Examples

Assume the SQL buffer contains the following query:

```
1  SELECT ENAME, DEPTNO
2  FROM EMP
3  WHERE JOB = 'SALESMAN'
4* ORDER BY DEPTNO
```

To make the line containing the WHERE clause the current line, you could enter

```
SQL> LIST 3
      3* WHERE JOB = 'SALESMAN'
```

followed by

```
SQL> DEL
```

The SQL buffer now contains the following lines:

```
1  SELECT ENAME, DEPTNO
2  FROM EMP
3* ORDER BY DEPTNO
```

To delete the second line of the buffer, enter

```
SQL> DEL 2
```

The SQL buffer now contains the following lines:

```
1  SELECT ENAME, DEPTNO
2* ORDER BY DEPTNO
```

DESCRIBE

Purpose

Lists the column definitions for the specified table, view, or synonym or the specifications for the specified function or procedure.

Syntax

```
DESC[RIBE] { [schema.] object[@net_service_name] }
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

schema

Represents the schema where the *object* resides. If you omit *schema*, SQL*Plus assumes you own *object*.

object

Represents the table, view, type, procedure, function, package or synonym you wish to describe.

@net_service_name

Consists of the database link name corresponding to the database where *object* exists. For more information on which privileges allow access to another table in a different schema, refer to the *Oracle8i SQL Reference*.

Usage Notes

The description for tables, views, types and synonyms contains the following information:

- each column's name
- whether or not null values are allowed (NULL or NOT NULL) for each column
- datatype of columns, for example, NUMBER, CHAR, VARCHAR2 (VARCHAR), LONG, DATE, RAW, LONGRAW, or ROWID
- precision of columns (and scale, if any, for a numeric column)

When you do a DESCRIBE, VARCHAR columns are returned with a type of VARCHAR2.

The DESCRIBE command allows you to describe objects recursively to the depth level set in the SET DESCRIBE command. You can also display the line number and indentation of the attribute or column name when an object contains multiple object types. For more information, see the SET command later in this chapter.

To control the width of the data displayed, use the SET LINESIZE command. For more information, see the SET command later in this chapter.

The description for functions and procedures contains the following information:

- the type of PL/SQL object (function or procedure)
- the name of the function or procedure
- the type of value returned (for functions)
- the argument names, types, whether they are input or output, and default values, if any

Examples

To describe the table EMP, enter

```
SQL> DESCRIBE EMP
```

SQL*Plus lists the following information:

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME		CHAR(10)
JOB		JOB(9)
MGR		NUMBER(4)
HIREDATE		DATE
SAL		NUMBER(7,2)
COMM		NUMBER(7,2)
DEPTNO		NUMBER(2)

To describe a procedure called CUSTOMER_LOOKUP, enter

```
SQL> DESCRIBE customer_lookup
```

SQL*Plus lists the following information:

```
PROCEDURE customer_lookup
Argument Name      Type      In/Out  Default?
-----
CUST_ID            NUMBER    IN
CUST_NAME          VARCHAR2  OUT
```

To create and describe the package APACK that contains the procedures aproc and bproc, enter

```
SQL> CREATE PACKAGE apack AS
  2  PROCEDURE aproc(P1 CHAR, P2 NUMBER);
  3  PROCEDURE bproc(P1 CHAR, P2 NUMBER);
  4  END apack;
  5  /
SQL> DESCRIBE apack
```

SQL*Plus lists the following information:

```
PROCEDURE aproc
Argument Name      Type      In/Out  Default?
-----
P1                 CHAR      IN
P2                 NUMBER    IN
PROCEDURE bproc
```

Argument Name	Type	In/Out	Default?
P1	CHAR	IN	
P2	NUMBER	IN	

To create and describe the object type ADDRESS that contains the attributes STREET and CITY, enter

```
SQL> CREATE TYPE ADDRESS AS OBJECT
  2  ( STREET  VARCHAR2(20),
  3    CITY    VARCHAR2(20)
  4  );
  5  /
SQL> DESCRIBE address
```

SQL*Plus lists the following information:

Name	Null?	Type
STREET		VARCHAR2(20)
CITY		VARCHAR2(20)

To create and describe the object type EMPLOYEE that contains the attributes ENAME, EMPADDR, JOB and SAL, enter

```
SQL> CREATE TYPE EMPLOYEE AS OBJECT
  2  ( ENAME    VARCHAR2(30),
  3    EMPADDR  ADDRESS,
  4    JOB      VARCHAR2(20),
  5    SAL      NUMBER(7,2)
  6  );
  7  /
SQL> DESCRIBE employee
```

SQL*Plus lists the following information:

Name	Null?	Type
ENAME		VARCHAR2(30)
EMPADDR		ADDRESS
JOB		VARCHAR2(20)
SAL		NUMBER(7,2)

To create and describe the object type addr_type as a table of the object type ADDRESS, enter

```
SQL> CREATE TYPE addr_type IS TABLE OF ADDRESS;
  2  /
SQL> DESCRIBE addr_type
```

SQL*Plus lists the following information:

addr_type TABLE OF ADDRESS		
Name	Null?	Type

```

-----
STREET                                VARCHAR2 ( 20 )
CITY                                  VARCHAR2 ( 20 )

```

To create and describe the object type `addr_varray` as a varray of the object type `ADDRESS`, enter

```

SQL> CREATE TYPE addr_varray AS VARRAY(10) OF ADDRESS;
      2 /
SQL> DESCRIBE addr_varray

```

SQL*Plus lists the following information:

```

addr_varray VARRAY(10) OF ADDRESS
Name                               Null?    Type
-----
STREET                             VARCHAR2 ( 20 )
CITY                               VARCHAR2 ( 20 )

```

To create and describe the table `dept_emp` that contains the columns `DEPTNO`, `PERSON` and `LOC`, enter

```

SQL> CREATE TABLE dept_emp
      2 ( DEPTNO  NUMBER,
      3   PERSON  EMPLOYEE,
      4   LOC     NUMBER
      5 );
      6 /
SQL> DESCRIBE dept_emp

```

SQL*Plus lists the following information:

```

Name                               Null?    Type
-----
DEPTNO                             NUMBER
PERSON                             EMPLOYEE
LOC                                 NUMBER

```

To create and describe the object type `rational` that contains the attributes `NUMERATOR` and `DENOMINATOR`, and the METHOD `rational_order`, enter

```

SQL> CREATE OR REPLACE TYPE rational AS OBJECT
      2 ( NUMERATOR  NUMBER,
      3   DENOMINATOR NUMBER,
      4   MAP MEMBER FUNCTION rational_order -
>     RETURN DOUBLE PRECISION,
      5   PRAGMA RESTRICT_REFERENCES
      6   (rational_order, RNDS, WNDS, RNPS, WNPS) );
      7 /
SQL> CREATE OR REPLACE TYPE BODY rational AS OBJECT
      2 MAP MEMBER FUNCTION rational_order -
>     RETURN DOUBLE PRECISION IS
      3 BEGIN
      4     RETURN NUMERATOR/DENOMINATOR;
      5 END;
      6 END;

```

```
7 /
SQL> DESCRIBE rational
```

SQL*Plus lists the following information:

Name	Null?	Type
-----	-----	-----
NUMERATOR		NUMBER
DENOMINATOR		NUMBER
METHOD		

MAP MEMBER FUNCTION RATIONAL_ORDER RETURNS NUMBER		

To describe the object emp_object and then format the output using the SET DESCRIBE command, first enter

```
SQL> desc emp_object
```

SQL*Plus lists the following information:

Name	Null	Type
-----	-----	-----
EMPLOYEE		RECUR_PERSON
DEPT		RECUR_DEPARTMENT
START_DATE		DATE
POSITION		VARCHAR2(1)
SAL		RECUR_SALARY

To format the DESCRIBE output use the SET command as follows:

```
SQL> set linesize 80
SQL> set desc depth 2
SQL> set desc indent on
SQL> set desc line off
```

To display the settings for the object, use the SHOW command as follows:

```
SQL> show desc
describe DEPTH 2 LINENUM OFF INDENT ON
SQL> desc emp_object
```

SQL*Plus lists the following information:

Name	Null	Type
-----	-----	-----
EMPLOYEE		RECUR_PERSON
NAME		VARCHAR2(20)
ADDR		RECUR_ADDRESS
ADDR1		RECUR_ADDRESS1
DOB		DATE
GENDER		VARCHAR2(10)
DEPT		RECUR_DEPARTMENT

DEPTNO	NUMBER
DEPT_NAME	VARCHAR2 (20)
LOCATION	VARCHAR2 (20)
START_DATE	DATE
POSITION	VARCHAR2 (1)
SAL	RECUR_SALARY
ANNUAL_SAL	NUMBER (10 , 2)
EMP_TYPE	VARCHAR2 (1)
COMM	NUMBER (10 , 2)
PENALTY_RATE	NUMBER (5 , 2)

For more information on using the CREATE TYPE command, see your *Oracle8i SQL Reference*.

For information about using the SET DESCRIBE and SHOW DESCRIBE commands, see the SET and SHOW commands later in this chapter.

DISCONNECT

Purpose

Commits pending changes to the database and logs the current username out of Oracle, but does not exit SQL*Plus.

Syntax

```
DISC[ONNECT]
```

Usage Notes

Use DISCONNECT within a command file to prevent user access to the database when you want to log the user out of Oracle but have the user remain in SQL*Plus. Use EXIT or QUIT to log out of Oracle and return control to your host computer's operating system.

Example

Your command file might begin with a CONNECT command and end with a DISCONNECT, as shown below.

```
SQL> GET MYFILE
  1  CONNECT ...
      .
      .
      .
15* DISCONNECT
```

EDIT

Purpose

Invokes a host operating system text editor on the contents of the specified file or on the contents of the buffer.

Syntax

```
ED[IT] [file_name[.ext]]
```

Terms and Clauses

Refer to the following for a description of the term or clause:

file_name[.ext]

Represents the file you wish to edit (typically a command file).

Enter EDIT with no filename to edit the contents of the SQL buffer with the host operating system text editor.

Usage Notes

If you omit the file extension, SQL*Plus assumes the default command-file extension (normally SQL). For information on changing the default extension, see the SUFFIX variable of the SET command in this chapter.

If you specify a filename, SQL*Plus searches for the file in the current working directory. If SQL*Plus cannot find the file in the current working directory, it creates a file with the specified name.

The user variable, _EDITOR, contains the name of the text editor invoked by EDIT. You can change the text editor by changing the value of _EDITOR. See DEFINE for information about changing the value of a user variable. If _EDITOR is undefined, EDIT attempts to invoke the default host operating system editor.

EDIT alone places the contents of the SQL buffer in a file by default named AFIEDT.BUF (in your current working directory) and invokes the text editor on the contents of the file. If the file AFIEDT.BUF already exists, it is overwritten with the contents of the buffer. You can change the default filename by using the SET EDITFILE command. For more information about setting a default filename for the EDIT command, see the EDITFILE variable of the SET command in this chapter.

Note:

The default file, AFIEDT.BUF, may have a different name on some operating systems.

If you do not specify a filename and the buffer is empty, EDIT returns an error message.

To leave the editing session and return to SQL*Plus, terminate the editing session in the way customary for the text editor. When you leave the editor, SQL*Plus loads the contents of the file into the buffer.

Example

To edit the file REPORT with the extension SQL using your host operating system text editor, enter

```
SQL> EDIT REPORT
```

EXECUTE

Purpose

Executes a single PL/SQL statement. The EXECUTE command is often useful when you want to execute a PL/SQL statement that references a stored procedure. For more information on PL/SQL, see your *PL/SQL User's Guide and Reference*.

Syntax

```
EXEC[UTE] statement
```

Terms and Clauses

Refer to the following for a description of the term or clause:

statement

Represents a PL/SQL statement.

Usage Notes

If your EXECUTE command cannot fit on one line because of the PL/SQL statement, use the SQL*Plus continuation character (a hyphen) as shown in the example below.

The length of the command and the PL/SQL statement cannot exceed the length defined by SET LINESIZE.

Examples

The following EXECUTE command assigns a value to a bind variable:

```
SQL> EXECUTE :n := 1
```

The following EXECUTE command runs a PL/SQL statement that references a stored procedure:

```
SQL> EXECUTE -  
> :ID := EMP_MANAGEMENT.HIRE('BLAKE','MANAGER','KING',2990,'SALES')
```


Note that the value returned by the stored procedure is being placed in a bind variable, :ID. For information on how to create a bind variable, see the VARIABLE command in this chapter.

EXIT

Purpose

Terminates SQL*Plus and returns control to the operating system.

Syntax

```
{EXIT|QUIT} [SUCCESS|FAILURE|WARNING|n|variable|:BindVariable] [COMMIT|ROLLBACK]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

{EXIT|QUIT}

Can be used interchangeably (QUIT is a synonym for EXIT).

SUCCESS

Exits normally.

FAILURE

Exits with a return code indicating failure.

WARNING

Exits with a return code indicating warning.

COMMIT

Saves pending changes to the database before exiting.

n

Represents an integer you specify as the return code.

variable

Represents a user-defined or system variable (but not a bind variable), such as SQL.SQLCODE. EXIT *variable* exits with the value of *variable* as the return code.

:BindVariable

Represents a variable created in SQL*Plus with the VARIABLE command, and then referenced in PL/SQL, or other subprograms. :BindVariable exits the subprogram and returns you to SQL*Plus.

ROLLBACK

Executes a ROLLBACK statement and abandons pending changes to the database before exiting.

EXIT with no clauses commits and exits with a value of SUCCESS.

Usage Notes

EXIT allows you to specify an operating system return code. This allows you to run SQL*Plus command files in batch mode and to detect programmatically the occurrence of an unexpected event. The manner of detection is operating-system specific. See the Oracle installation and user's manual(s) provided for your operating system for details.

The key words SUCCESS, WARNING, and FAILURE represent operating-system dependent values. On some systems, WARNING and FAILURE may be indistinguishable.

Note:

SUCCESS, FAILURE, and WARNING are not reserved words.

The range of operating system return codes is also restricted on some operating systems. This limits the portability of EXIT *n* and EXIT *variable* between platforms. For example, on UNIX there is only one byte of storage for return codes; therefore, the range for return codes is limited to zero to 255.

If you make a syntax error in the EXIT options or use a non-numeric variable, SQL*Plus performs an EXIT FAILURE COMMIT.

For information on exiting conditionally, see the WHENEVER SQLERROR and WHENEVER OSERROR commands later in this chapter.

Example

The following example commits all uncommitted transactions and returns the error code of the last executed SQL command or PL/SQL block:

```
SQL> EXIT SQL.SQLCODE
```

The location of the return code depends on your system. Consult your DBA for information concerning how your operating system retrieves data from a program. See the TTITLE command in this chapter for more information on SQL.SQLCODE.

GET

Purpose

Loads a host operating system file into the SQL buffer.

Syntax

```
GET file_name[.ext] [LIS[T]|NOL[IST]]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

***file_name*[.ext]**

Represents the file you wish to load (typically a command file).

LIS[T]

Lists the contents of the file.

NOL[IST]

Suppresses the listing.

Usage Note

If you do not specify a file extension, SQL*Plus assumes the default command-file extension (normally SQL). For information on changing the default extension, see the SUFFIX variable of the SET command in this chapter.

If part of the filename you are specifying contains the word *list* or the word *file*, you need to put the name in double quotes.

SQL*Plus searches for the file in the current working directory.

The operating system file should contain a single SQL statement or PL/SQL block. The statement should not be terminated with a semicolon.

If a SQL*Plus command or more than one SQL statement or PL/SQL block is loaded into the SQL buffer from an operating system file, an error occurs when the RUN or slash (/) command is used to execute the buffer.

The GET command can be used to load files created with the SAVE command. See the [SAVE](#) command in this chapter for more information.

Example

To load a file called YEARENDRPT with the extension SQL into the buffer, enter

```
SQL> GET YEARENDRPT
```

HELP

Purpose

Accesses the SQL*Plus help system.

Syntax

```
HELP [topic]
```

Terms and Clauses

Refer to the following for a description of the term or clause:

topic

Represents a SQL*Plus help topic, for example, COLUMN.

Enter HELP without *topic* to get help on the help system.

Usage Notes

You can only enter one topic after HELP. You can abbreviate the topic (for example, COL for COLUMN). However, if you enter only an abbreviated topic and the abbreviation is ambiguous, SQL*Plus displays help for all topics that match the abbreviation. For example, if you enter

```
SQL> HELP EX
```

SQL*Plus displays the syntax for the EXECUTE command followed by the syntax for the EXIT command.

If you get a response indicating that help is not available, consult your database administrator.

Example

To see a list of SQL*Plus commands, enter

```
SQL> HELP INDEX
```

HOST

Purpose

Executes a host operating system command without leaving SQL*Plus.

Syntax

HO[ST] [*command*]

Terms and Clauses

Refer to the following for a description of the term or clause:

command

Represents a host operating system command.

Enter HOST without *command* to display an operating system prompt. You can then enter multiple operating system commands. For information on returning to SQL*Plus, refer to the Oracle installation and user's manual(s) provided for your operating system.

Usage Notes

With some operating systems, you can use a "\$" (VMS), "!" (UNIX), or another character instead of HOST. See the Oracle installation and user's manual(s) provided for your operating system for details.

You may not have access to the HOST command, depending on your operating system. See the Oracle installation and user's manual(s) provided for your operating system or ask your DBA for more information.

SQL*Plus removes the SQLTERMINATOR (a semicolon by default) before the HOST command is issued. A workaround for this is to add another SQLTERMINATOR. See the SQLTERMINATOR variable of the SET command in this chapter for more information on the SQLTERMINATOR.

Example

To execute an operating system command, ls *.sql, enter

```
SQL> HOST ls *.sql
```

INPUT

Purpose

Adds one or more new lines of text after the current line in the buffer.

Syntax

I[INPUT] [*text*]

Terms and Clauses

Refer to the following for a description of the term or clause:

text

Represents the text you wish to add. To add a single line, enter the text of the line after the command INPUT, separating the text from the command with a space. To begin the line with one or more spaces, enter two or more spaces between INPUT and the first non-blank character of *text*.

To add several lines, enter INPUT with no *text*. INPUT prompts you for each line. To leave INPUT, enter a null (empty) line.

Usage Notes

If you enter a line number at the command prompt larger than the number of lines in the buffer, and follow the number with text, SQL*Plus adds the text in a new line at the end of the buffer. If you specify zero (0) for the line number and follow the zero with text, then SQL*Plus inserts the line at the beginning of the buffer (that line becomes line 1).

Examples

Assume the SQL buffer contains the following command:

```
1  SELECT ENAME, DEPTNO, SAL, COMM
2  FROM EMP
```

To add an ORDER BY clause to the query, enter

```
SQL> LIST 2
      2* FROM EMP
SQL> INPUT ORDER BY ENAME
```

LIST 2 ensures that line 2 is the current line. INPUT adds a new line containing the ORDER BY clause after the current line. The SQL buffer now contains the following lines:

```
1  SELECT ENAME, DEPTNO, SAL, COMM
2  FROM EMP
3* ORDER BY ENAME
```

To add a two-line WHERE clause, enter

```
SQL> LIST 2
      2* FROM EMP
SQL> INPUT
      3 WHERE JOB = 'SALESMAN'
      4 AND COMM 500
      5
```

INPUT prompts you for new lines until you enter an empty line. The SQL buffer now contains the following lines:

```
1  SELECT ENAME, DEPTNO, SAL, COMM
2  FROM EMP
3  WHERE JOB = 'SALESMAN'
4  AND COMM 500
5  ORDER BY ENAME
```

LIST

Purpose

Lists one or more lines of the SQL buffer.

Syntax

`L[IST] [n|n m|n *|n LAST|*|* n|* LAST|LAST]`

Terms and Clauses

Refer to the following list for a description of each term or clause:

n	Lists line <i>n</i> .
n m	Lists lines <i>n</i> through <i>m</i> .
n *	Lists line <i>n</i> through the current line.
n LAST	Lists line <i>n</i> through the last line.
*	Lists the current line.
* n	Lists the current line through line <i>n</i> .
* LAST	Lists the current line through the last line.
LAST	Lists the last line.

Enter LIST with no clauses to list all lines.

Usage Notes

The last line listed becomes the new current line (marked by an asterisk).

Example

To list the contents of the buffer, enter

```
SQL> LIST
```

You will see a listing of all lines in the buffer, similar to the following example:

```
1  SELECT ENAME, DEPTNO, JOB
2  FROM EMP
3  WHERE JOB = 'CLERK'
4* ORDER BY DEPTNO
```

The asterisk indicates that line 4 is the current line.

To list the second line only, enter

```
SQL> LIST 2
```

You will then see this:

```
2* FROM EMP
```

To list the current line (now line 2) to the last line, enter

```
SQL> LIST * LAST
```

You will then see this:

```
2  FROM EMP
3  WHERE JOB = 'CLERK'
4* ORDER BY DEPTNO
```

PASSWORD

Purpose

Allows you to change a password without echoing the password on an input device.

Syntax

```
PASSW[ORD] [username]
```

Terms and Clauses

Refer to the following for a description of the clause or term:

username

Specifies the user. If you do not specify a username, *username* defaults to the current user.

Usage Notes

To change the password of another user, you must have been granted the appropriate privilege.

For more information about changing your password, see the CONNECT command in this chapter.

Example

Suppose you are logged on as scott/tiger, and want to change the password to tigertiger

```
SQL> passw
Changing password for scott
Old password: tiger
New password: tigertiger
Retype new password: tigertiger
Password changed
```

Suppose you are logged on as a DBA, and want to change the password for user usera (currently identified by passa) to passusera

```
SQL> passw usera
Changing password for usera
New password: passusera
Retype new password: passusera
Password changed
```

PAUSE

Purpose

Displays an empty line followed by a line containing text, then waits for the user to press [Return], or displays two empty lines and waits for the user's response.

Syntax

```
PAU[SE] [text]
```

Terms and Clauses

Refer to the following for a description of the clause or term:

text

Represents the text you wish to display.

Enter PAUSE followed by no text to display two empty lines.

Usage Notes

Because PAUSE always waits for the user's response, it is best to use a message that tells the user explicitly to press [Return].

PAUSE reads input from the terminal (if a terminal is available) even when you have designated the source of the command input as a file.

For information on pausing between pages of a report, see the PAUSE variable of the SET command later in this chapter.

Example

To print "Adjust paper and press RETURN to continue." and to have SQL*Plus wait for the user to press [Return], you might include the following PAUSE command in a command file:

```
SET PAUSE OFF
PAUSE Adjust paper and press RETURN to continue.
SELECT ...
```

PRINT

Purpose

Displays the current value of bind variables. For more information on bind variables, see your *PL/SQL User's Guide and Reference*.

Syntax

```
PRI[NT] [variable ...]
```

Terms and Clauses

Refer to the following for a description of the clause or term:

variable ...

Represents the names of the bind variables whose values you wish to display.

Enter PRINT with no variables to print all bind variables.

Usage Notes

Bind variables are created using the VARIABLE command. For more information and examples, see the VARIABLE command in this chapter.

You can control the formatting of the PRINT output just as you would query output. For more information, see the formatting techniques described in Chapter 4.

To automatically display bind variables referenced in a successful PL/SQL block or used in an EXECUTE command, use the AUTOPRINT clause of the SET command. For more information, see the SET command in this chapter.

Example

The following example illustrates a PRINT command:

```
SQL> VARIABLE n NUMBER
SQL> BEGIN
  2   :n := 1;
  3   END;
SQL> PRINT n
      N
-----
      1
```

PROMPT

Purpose

Sends the specified message or a blank line to the user's screen.

Syntax

```
PRO[MPT] [text]
```

Terms and Clauses

Refer to the following for a description of the term or clause:

text

Represents the text of the message you wish to display. If you omit *text*, PROMPT displays a blank line on the user's screen.

Usage Notes

You can use this command in command files to give information to the user.

Example

The following example shows the use of PROMPT in conjunction with ACCEPT in a command file called ASKFORDEPT. ASKFORDEPT contains the following SQL*Plus and SQL commands:

```
PROMPT
PROMPT Please enter a valid department
PROMPT For example:  10, 20, 30, 40
ACCEPT NEWDEPT NUMBER PROMPT 'DEPT:> '
SELECT DNAME FROM DEPT
WHERE DEPTNO = &NEWDEPT
```

Assume you run the file using START or @:

```
SQL> @ASKFORDEPT
```

SQL*Plus displays the following prompts:

```
Please enter a valid department
```

For example: 10, 20, 30, 40
DEPT:>

You can enter a department number at the prompt DEPT:>. By default, SQL*Plus lists the line containing &NEWDEPT before and after substitution, and then displays the department name corresponding to the number entered at the DEPT:> prompt.

RECOVER

Purpose

Performs media recovery on one or more tablespaces, one or more datafiles, or the entire database.

Syntax

```
RECOVER [AUTOMATIC][FROM location]
  {[STANDBY] DATABASE [UNTIL options][USING BACKUP CONTROLFILE]
  |TABLESPACE {tablespace [, tablespace ...]}
  |DATAFILE {datafilename [, datafilename ...]}
  |STANDBY {TABLESPACE tablespace [,tablespace ...]
  |DATAFILE datafilename [, datafilename ...]} UNTIL CONTROLFILE
  |LOGFILE filename
  |CONTINUE [DEFAULT]
  |CANCEL}
[PARALLEL clause]
```

where *options* requires the following syntax:

```
{CANCEL|CHANGE integer|TIME date}
```

and where *clause* requires the following syntax:

```
{PARALLEL ([DEGREE {integer|DEFAULT}|INSTANCES {integer|DEFAULT}]...)
|NOPARALLEL}
```

Terms and Clauses

Refer to the following list for a description of each term and clause:

AUTOMATIC

Automatically generates the name of the next archived redo log file needed to continue the recovery operation. Oracle uses the LOG_ARCHIVE_DEST (or LOG_ARCHIVE_DEST_1) and LOG_ARCHIVE_FORMAT parameters (or their defaults) to generate the target redo log filename. If the file is found, the redo contained in that file is applied. If the file is not found, Oracle prompts you for a filename, displaying the generated filename as a suggestion.

If you specify neither AUTOMATIC nor LOGFILE, Oracle prompts you for a filename, displaying the generated filename as a suggestion. You can then accept the

generated filename or replace it with a fully qualified filename. If you know the archived filename differs from what Oracle would generate, you can save time by using the LOGFILE clause.

FROM *location*

Specifies the location from which the archived redo log file group is read. The value of location must be a fully specified file location following the conventions of your operating system. If you omit this parameter, Oracle assumes the archived redo log file group is in the location specified by the initialization parameter LOG_ARCHIVE_DEST or LOG_ARCHIVE_DEST_1.

STANDBY

Recovers the standby database using the control file and archived redo log files copied from the primary database. The standby database must be mounted but not open.

DATABASE

Recovers the entire database.

UNTIL CANCEL

Specifies an incomplete, cancel-based recovery. Recovery proceeds by prompting you with the suggested filenames of archived redo log files, and recovery completes when you specify CANCEL instead of a filename.

UNTIL CHANGE *integer*

Specifies an incomplete, change-based recovery. *integer* is the number of the System Change Number (SCN) following the last change you wish to recover. For example, if you want to restore your database up to the transaction with an SCN of 9, you would specify UNTIL CHANGE 10.

UNTIL TIME *date*

Specifies an incomplete, time-based recovery. Use single quotes, and the following format:

'YYYY-MM-DD:HH24:MI:SS'

USING BACKUP CONTROLFILE

Specifies that a backup of the control file be used instead of the current control file.

TABLESPACE *tablespace*

Recovers a particular tablespace. *tablespace* is the name of a tablespace in the current database. You may recover up to 16 tablespaces in one statement.

DATAFILE *datafilename*

Recovers a particular datafile. You can specify any number of datafiles.

STANDBY {**TABLESPACE** *tablespace* [, *tablespace* ...]}

[**DATAFILE** *datafilename* [, *datafilename* ...]]

Reconstructs a lost or damaged datafile or tablespace in the standby database using archived redo log files copied from the primary database and a control file.

UNTIL CONTROLFILE

Specifies that the recovery of an old standby datafile or tablespace uses the current standby database control file.

LOGFILE *filename*

Continues media recovery by applying the specified redo log file.

CONTINUE [**DEFAULT**]

Continues multi-instance recovery after it has been interrupted to disable a thread.

Continues recovery using the redo log file that Oracle would automatically generate if no other logfile were specified. This option is equivalent to specifying **AUTOMATIC**, except that Oracle does not prompt for a filename.

CANCEL

Terminates cancel-based recovery.

PARALLEL DEGREE *integer*

Specifies the number of recovery processes used to apply redo entries to datafiles on each instance. An integer specified for **DEGREE** overrides the initialization parameter **RECOVERY_PARALLELISM**.

PARALLEL DEGREE DEFAULT

Indicates that twice the number of datafiles being recovered is the number of recovery processes to use.

PARALLEL INSTANCES *integer*

Specifies the number of instances to use for parallel recovery.

The number of recovery processes specified with **DEGREE** is used on each instance. Thus, the total number of recovery processes is the integer specified with **DEGREE** multiplied by the integer specified with **INSTANCES**. **INSTANCES** is only pertinent for the Oracle Parallel Server.

PARALLEL INSTANCES DEFAULT

INSTANCES DEFAULT or not including the **INSTANCES** keyword causes has operating system-specific consequences. For more information about the default

behavior of the INSTANCES DEFAULT specification, see the *Oracle8i Parallel Server Concepts and Administration* manual.

NOPARALLEL

Specifies that recovery is to proceed serially. Note that a specification of PARALLEL(DEGREE 1 INSTANCES 1) is equivalent to specifying the NOPARALLEL keyword.

The PARALLEL keyword overrides the RECOVERY_PARALLELISM initialization parameter. The number specified with the PARALLEL keyword is the number of recovery processes used to apply redo entries to datafiles. For more information about the PARALLEL keyword see the *Oracle8i Parallel Server Concepts and Administration* manual.

Usage Notes

Note, you must be connected to Oracle as SYSOPER, or SYSDBA. You cannot use the RECOVER command when connected via the multi-threaded server.

To perform media recovery on an entire database (all tablespaces), the database must be mounted EXCLUSIVE and closed.

To perform media recovery on a tablespace, the database must be mounted and open, and the tablespace must be offline.

To perform media recovery on a datafile, the database can remain open and mounted with the damaged datafiles offline (unless the file is part of the SYSTEM tablespace).

Before using the RECOVER command you must have restored copies of the damaged datafile(s) from a previous backup. Be sure you can access all archived and online redo log files dating back to when that backup was made.

When another log file is required during recovery, a prompt suggests the names of files that are needed. The name is derived from the values specified in the initialization parameters LOG_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT. You should restore copies of the archived redo log files needed for recovery to the destination specified in LOG_ARCHIVE_DEST, if necessary. You can override the initialization parameters by setting the LOGSOURCE variable with the SET LOGSOURCE command.

During recovery you can accept the suggested log name by pressing return, cancel recovery by entering CANCEL instead of a log name, or enter AUTO at the prompt for automatic file selection without further prompting.

If you have enabled autorecovery (that is, SET AUTORECOVERY ON), recovery proceeds without prompting you with filenames. Status messages are displayed when each log file is applied.

When normal media recovery is done, a completion status is returned.

For more information on recovery and the RECOVER command, see the *Oracle8i Administrator's Guide*, and the *Oracle8i Backup and Recovery* guide.

Examples

To recover the entire database, enter

```
SQL> RECOVER DATABASE
```

To recover the database until a specified time, enter

```
SQL> RECOVER DATABASE UNTIL TIME 23-NOV-98:04:32:00
```

To recover the two tablespaces ts_one and ts_two from the database, enter

```
SQL> RECOVER TABLESPACE ts_one, ts_two
```

To recover the datafile data1.db from the database, enter

```
SQL> RECOVER DATAFILE 'data1.db'
```

REMARK

Purpose

Begins a comment in a command file. SQL*Plus does not interpret the comment as a command.

Syntax

```
REM[ARK]
```

Usage Notes

The REMARK command must appear at the beginning of a line, and the comment ends at the end of the line. A line cannot contain both a comment and a command.

For details on entering comments in command files using the SQL comment delimiters, /* ... */ , or the ANSI/ISO comment delimiter, - - ..., refer to ["Placing Comments in Command Files"](#) in [Chapter 3](#).

Example

The following command file contains some typical comments:

```
REM COMPUTE uses BREAK ON REPORT to break on end of table
BREAK ON REPORT
COMPUTE SUM OF "DEPARTMENT 10" "DEPARTMENT 20" -
"DEPARTMENT 30" "TOTAL BY JOB" ON REPORT
REM Each column displays the sums of salaries by job for
REM one of the departments 10, 20, 30.
SELECT JOB,
        SUM( DECODE( DEPTNO, 10, SAL, 0)) "DEPARTMENT 10",
        SUM( DECODE( DEPTNO, 20, SAL, 0)) "DEPARTMENT 20",
        SUM( DECODE( DEPTNO, 30, SAL, 0)) "DEPARTMENT 30",
        SUM(SAL) "TOTAL BY JOB"
```


FROM EMP
GROUP BY JOB

REPFOOTER

Purpose

Places and formats a specified report footer at the bottom of each report, or lists the current REPFOOTER definition.

Syntax

```
REPF[OOTER] [PAGE] [printspec [text|variable] ...] | [OFF|ON]
```

Terms and Clauses

Refer to the [REPHEADER](#) command for additional information on terms and clauses in the REPFOOTER command syntax.

Enter REPFOOTER with no clauses to list the current REPFOOTER definition.

Usage Notes

If you do not enter a *printspec* clause before the text or variables, REPFOOTER left justifies the text or variables.

You can use any number of constants and variables in a *printspec*. SQL*Plus displays the constants and variables in the order you specify them, positioning and formatting each constant or variable as specified by the *printspec* clauses that precede it.

Note:

If SET EMBEDDED is ON, the report footer is suppressed.

Example

To define "END EMPLOYEE LISTING REPORT" as a report footer on a separate page and to center it, enter:

```
SQL> REPFOOTER PAGE CENTER 'END EMPLOYEE LISTING REPORT'  
SQL> TTITLE RIGHT 'Page: ' FORMAT 999 SQL.PNO  
SQL> SELECT ENAME, SAL  
2 FROM EMP  
3 WHERE SAL > 2000;
```

Page: 1

ENAME	SAL
-----	-----
JONES	2975
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
FORD	3000

Page: 2

END EMPLOYEE LISTING REPORT

To suppress the report footer without changing its definition, enter

```
SQL> REPFOOTER OFF
```

REPHEADER

Purpose

Places and formats a specified report header at the top of each report, or lists the current REPHEADER definition.

Syntax

```
REPH[EADER] [PAGE] [printspec [text|variable] ...] | [OFF|ON]
```

where *printspec* represents one or more of the following clauses used to place and format the *text*:

```
COL n
S[KIP] [n]
TAB n
LE[FT]
CE[NTER]
R[IGHT]
BOLD
FORMAT text
```

Terms and Clauses

Refer to the following list for a description of each term or clause. These terms and clauses also apply to the REPFOOTER command.

PAGE

Begins a new page after printing the specified report header or before printing the specified report footer.

text

Represents the report header or footer text. Enter *text* in single quotes if you want to place more than one word on a single line. The default is NULL.

variable

Represents a user variable or any of the following system-maintained values:

- SQL.LNO (current line number)
- SQL.PNO (current page number)
- SQL.RELEASE (current Oracle release number)
- SQL.CODE (current error code)
- SQL.USER (current username)

To print one of these values, reference the appropriate variable in the report header or footer. You can format *variable* with the FORMAT clause.

OFF

Turns the report header or footer off (suppresses its display) without affecting its definition.

COL *n*

Indents to column *n* of the current line (backward if column *n* has been passed). "Column" in this context means print position, not table column.

S[KIP] [*n*]

Skips to the start of a new line *n* times; if you omit *n*, one time; if you enter zero for *n*, backward to the start of the current line.

TAB *n*

Skips forward *n* columns (backward if you enter a negative value for *n*). "Column" in this context means print position, not table column.

LE[FT] CE[NTER] R[IGHT]

Left-align, center, and right-align data on the current line respectively. SQL*Plus aligns following data items as a group, up to the end of the *printspec* or the next LEFT, CENTER, RIGHT, or COL command. CENTER and RIGHT use the SET LINESIZE value to calculate the position of the data item that follows.

BOLD

Prints data in bold print. SQL*Plus represents bold print on your terminal by repeating the data on three consecutive lines. On some operating systems, SQL*Plus may instruct your printer to print bolded text on three consecutive lines, instead of bold.

FORMAT *text*

Specifies a format model that determines the format of following data items, up to the next FORMAT clause or the end of the command. The format

model must be a *text* constant such as A10 or \$999. See **COLUMN FORMAT** for more information on formatting and valid format models.

If the datatype of the format model does not match the datatype of a given data item, the **FORMAT** clause has no effect on that item.

If no appropriate **FORMAT** model precedes a given data item, **SQL*Plus** prints **NUMBER** values according to the format specified by **SET NUMFORMAT** or, if you have not used **SET NUMFORMAT**, the default format. **SQL*Plus** prints **DATE** values according to the default format.

Refer to the **FORMAT** clause of the **COLUMN** command in this chapter for more information on default formats.

Enter **REPHEADER** with no clauses to list the current **REPHEADER** definition.

Usage Notes

If you do not enter a *printspec* clause before the text or variables, **REPHEADER** left justifies the text or variables.

You can use any number of constants and variables in a *printspec*. **SQL*Plus** displays the constants and variables in the order you specify them, positioning and formatting each constant or variable as specified by the *printspec* clauses that precede it.

Example

To define "EMPLOYEE LISTING REPORT" as a report header on a separate page, and to center it, enter:

```
SQL> REPHEADER PAGE CENTER 'EMPLOYEE LISTING REPORT'
SQL> TTITLE RIGHT 'Page: ' FORMAT 999 SQL.PNO
SQL> SELECT ENAME, SAL
      2 FROM EMP
      3 WHERE SAL > 2000;
```

Page: 1

EMPLOYEE LISTING REPORT

Page: 2

ENAME	SAL
JONES	2975
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
FORD	3000

6 rows selected.

To suppress the report header without changing its definition, enter:

```
SQL> REPHEADER OFF
```

RUN

Purpose

Lists and executes the SQL command or PL/SQL block currently stored in the SQL buffer.

Syntax

```
R[UN]
```

Usage Notes

RUN causes the last line of the SQL buffer to become the current line.

The slash command (/) functions similarly to RUN, but does not list the command in the SQL buffer on your screen.

Example

Assume the SQL buffer contains the following query:

```
SELECT DEPTNO FROM DEPT
```

To RUN the query, enter

```
SQL> RUN
```

The following output results:

```
1* SELECT DEPTNO FROM DEPT
```

```
DEPTNO
-----
      10
      20
      30
      40
```

SAVE

Purpose

Saves the contents of the SQL buffer in a host operating system file (a command file).

Syntax

```
SAV[E] file_name[.ext] [CRE[ATE] | REP[LACE] | APP[END]]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

file_name[.ext]

Specifies the command file in which you wish to save the buffer's contents.

REP[LACE]

Replaces the contents of an existing file. If the file does not exist, REPLACE creates the file.

APP[END]

Adds the contents of the buffer to the end of the file you specify.

Usage Notes

If you do not specify an extension, SQL*Plus assumes the default command-file extension (normally SQL). For information on changing this default extension, see the SUFFIX variable of the SET command in this chapter.

If you wish to SAVE a file under a name identical to a SAVE command clause (CREATE, REPLACE, or APPEND), you must specify a file extension.

When you SAVE the contents of the SQL buffer, SAVE adds a line containing a slash (/) to the end of the file.

If the filename you specify is the word *file*, you need to put the name in single quotes.

Examples

To save the contents of the buffer in a file named DEPTSALRPT with the extension SQL, enter

```
SQL> SAVE DEPTSALRPT
```

To save the contents of the buffer in a file named DEPTSALRPT with the extension OLD, enter

```
SQL> SAVE DEPTSALRPT.OLD
```

SET

Purpose

Sets a system variable to alter the SQL*Plus environment for your current session, such as

- the display width for NUMBER data
- the display width for LONG data
- enabling or disabling the printing of column headings
- the number of lines per page

Syntax

SET *system_variable* *value*

where *system_variable value* represents a system variable followed by a value, as shown below:

```

APPI[NFO] {ON|OFF|text}
ARRAY[SIZE] {15|n}
AUTO[COMMIT] {OFF|ON|IMM[EDIATE]|n}
AUTOP[RINT] {OFF|ON}
AUTORECOVERY {ON|OFF}
AUTOT[RACE] {OFF|ON|TRACE[ONLY]} [EXP[LAIN]] [STAT[ISTICS]]
BLO[CKTERMINATOR] {.c}
CMDS[EP] {;c|OFF|ON}
COLSEP {_|text}
COM[PATIBILITY] {V7|V8|NATIVE}
CON[CAT] {.c|OFF|ON}
COPYC[OMMIT] {0|n}
COPYTYPECHECK {OFF|ON}
DEF[INE] {'&'|c|OFF|ON}
DESCRIBE [DEPTH {1|n|ALL}][LINENUM {ON|OFF}][INDENT {ON|OFF}]
ECHO {OFF|ON}
EDITF[ILE] file_name [.ext]
EMB[EDDED] {OFF|ON}
ESC[APE] {\|c|OFF|ON}
FEED[BACK] {6|n|OFF|ON}
FLAGGER {OFF|ENTRY|INTERMED[iate]|FULL}
FLU[SH] {OFF|ON}
HEA[DING] {OFF|ON}
HEADS[EP] {1|c|OFF|ON}
INSTANCE [instance_path|LOCAL]
LIN[ESIZE] {80|n}
LOBOF[FSET] {n|1}
LOGSOURCE [pathname]
LONG {80|n}
LONGC[HUNKSIZE] {80|n}
NEWP[AGE] {1|n|NONE}
NULL text
NUMF[ORMAT] format
NUM[WIDTH] {10|n}
PAGES[IZE] {24|n}
PAU[SE] {OFF|ON|text}
RECSEP {WR[APPED]|EA[CH]|OFF}
RECSEPCHAR {_|c}
SERVEROUT[PUT] {OFF|ON} [SIZE n] [FOR[MAT] {WRA[PPED]|
    WOR[D_WRAPPED]|TRU[NCATED]}]
SHIFT[INOUT] {VIS[IBLE]|INV[ISIBLE]}
SHOW[MODE] {OFF|ON}
SQLBL[ANKLINES] {ON|OFF}
SQLC[ASE] {MIX[ED]|LO[WER]|UP[PER]}
SQLCO[NTINUE] {>|text}
SQLN[UMBER] {OFF|ON}
SQLPRE[FIX] {#|c}
SQLP[ROMPT] {SQL>|text}
SQLT[ERMINATOR] {;c|OFF|ON}
SUF[IX] {SQL|text}
TAB {OFF|ON}
TERM[OUT] {OFF|ON}
TI[ME] {OFF|ON}
TIMI[NG] {OFF|ON}
TRIM[OUT] {OFF|ON}
TRIMS[POOL] {ON|OFF}

```

UND[ERLINE] {-|c|ON|OFF}
 VER[IFY] {OFF|ON}
 WRA[P] {OFF|ON}

Terms and Clauses

Refer to the following list for a description of each term, clause, or system variable:

APPI[NFO]{ON|OFF|*text*}

Sets automatic registering of command files through the DBMS_APPLICATION_INFO package. This enables the performance and resource usage of each command file to be monitored by your DBA. The registered name appears in the MODULE column of the V\$SESSION and V\$SQLAREA virtual tables. You can also read the registered name using the DBMS_APPLICATION_INFO.READ_MODULE procedure.

ON registers command files invoked by the @, @@ or START commands. OFF disables registering of command files. Instead, the current value of *text* is registered. *Text* specifies the text to register when no command file is being run or when APPINFO is OFF. The default for *text* is "SQL*Plus". If you enter multiple words for *text*, you must enclose them in quotes. The maximum length for *text* is limited by the DBMS_APPLICATION_INFO package.

The registered name has the format *nn@xfilename* where: *nn* is the depth level of command file; *x* is '<' when the command file name is truncated, otherwise, it is blank; and *filename* is the command file name, possibly truncated to the length allowed by the DBMS_APPLICATION_INFO package interface.

Note:

To use this feature, you must have access to the DBMS_APPLICATION_INFO package. Run DBMSUTIL.SQL (this name may vary depending on your operating system) as SYS to create the DBMS_APPLICATION_INFO package. DBMSUTIL.SQL is part of the Oracle8 database server product.

For more information on the DBMS_APPLICATION_INFO package, see the *Oracle8i Tuning* manual.

ARRAY[SIZE] {15|*n*}

Sets the number of rows--called a *batch*--that SQL*Plus will fetch from the database at one time. Valid values are 1 to 5000. A large value increases the efficiency of queries and subqueries that fetch many rows, but requires more memory. Values over approximately 100 provide little added performance. ARRAYSIZE has no effect on the results of SQL*Plus operations other than increasing efficiency.

AUTO[COMMIT] {OFF|ON|IMM[EDIATE] | *n*}

Controls when Oracle commits pending changes to the database. ON commits pending changes to the database after Oracle executes each successful INSERT, UPDATE, or DELETE command or PL/SQL block. OFF suppresses automatic committing so that you must commit changes manually (for example, with the SQL command COMMIT). IMMEDIATE functions in the same manner as the ON option. *n* commits pending changes to the database after Oracle executes *n* successful SQL INSERT, UPDATE, or DELETE commands or PL/SQL blocks. *n* cannot be less than zero or greater than 2,000,000,000. The statement counter is reset to zero after successful completion of

- *n* INSERT, UPDATE or DELETE commands or PL/SQL blocks
- a commit
- a rollback
- a SET AUTOCOMMIT command

Note:

For this feature, a PL/SQL block is considered one transaction, regardless of the actual number of SQL commands contained within it.

AUTOP[RINT] {OFF|ON}

Sets the automatic PRINTing of bind variables. ON or OFF controls whether SQL*Plus automatically displays bind variables (referenced in a successful PL/SQL block or used in an EXECUTE command). For more information about displaying bind variables, see the PRINT command in this chapter.

AUTORECOVERY [ON|OFF]

ON sets the RECOVER command to automatically apply the default filenames of archived redo log files needed during recovery. No interaction is needed when AUTORECOVERY is set to ON, provided the necessary files are in the expected locations with the expected names. The filenames used when AUTORECOVERY is ON are derived from the values of the initialization parameters LOG_ARCHIVE_DEST and LOG_ARCHIVE_FORMAT.

OFF, the default option, requires that you enter the filenames manually or accept the suggested default filename given.

AUTOT[RACE] {OFF|ON|TRACE[ONLY]} [EXP[LAIN]] [STAT[ISTICS]]

Displays a report on the execution of successful SQL DML statements (SELECT, INSERT, UPDATE or DELETE). The report can include execution statistics and the query execution path.

OFF does not display a trace report. ON displays a trace report. TRACEONLY displays a trace report, but does not print query data, if any. EXPLAIN shows the query execution path by performing an EXPLAIN PLAN. STATISTICS displays SQL statement statistics.

Using ON or TRACEONLY with no explicit options defaults to EXPLAIN STATISTICS.

The TRACEONLY option may be useful to suppress the query data of large queries. If STATISTICS is specified, SQL*Plus still fetches the query data from the server, however, the data is not displayed.

The AUTOTRACE report is printed after the statement has successfully completed.

Information about Execution Plans and the statistics is documented in the *Oracle8i Tuning* manual.

When SQL*Plus produces a STATISTICS report, a second connection to the database is automatically created. This connection is closed when the STATISTICS option is set to OFF, or you log out of SQL*Plus.

The formatting of your AUTOTRACE report may vary depending on the version of the server to which you are connected and the configuration of the server.

AUTOTRACE is not available when FIPS flagging is enabled.

See "[Tracing Statements](#)" in [Chapter 3](#) for more information on AUTOTRACE.

BLO[CKTERMINATOR] {**.**|**c**}

Sets the non-alphanumeric character used to end PL/SQL blocks to *c*. To execute the block, you must issue a RUN or / (slash) command.

CMDS[EP] {**:**|**c**|**OFF**|**ON**}

Sets the non-alphanumeric character used to separate multiple SQL*Plus commands entered on one line to *c*. ON or OFF controls whether you can enter multiple commands on a line; ON automatically sets the command separator character to a semicolon (;).

COLSEP { | *text* }

Sets the text to be printed between SELECTed columns. If the COLSEP variable contains blanks or punctuation characters, you must enclose it with single quotes. The default value for *text* is a single space.

In multi-line rows, the column separator does not print between columns that begin on different lines. The column separator does not appear on blank lines produced by BREAK ... SKIP *n* and does not overwrite the record separator. See [SET RECSEP](#) in this chapter for more information.

COM[patibility] {**v7**|**v8**|**NATIVE**}

Specifies the version of Oracle to which you are currently connected. Set COMPATIBILITY to V7 for Oracle7, or V8 for Oracle8 and Oracle8i. Set COMPATIBILITY to NATIVE if you wish the database to determine the setting (for example, if connected to Oracle8 or Oracle8i, compatibility would default to V8). COMPATIBILITY must be correctly set for the version of Oracle to which you are connected; otherwise, you will be unable to run any SQL commands. Note that you can set COMPATIBILITY to V7 when connected to Oracle8i. This enables you to run Oracle7 SQL against Oracle8i.

CON[CAT] {.*c*|OFF|ON}

Sets the character you can use to terminate a substitution variable reference if you wish to immediately follow the variable with a character that SQL*Plus would otherwise interpret as a part of the substitution variable name. SQL*Plus resets the value of CONCAT to a period when you switch CONCAT on.

COPYC[OMMIT] {0|*n*}

Controls the number of batches after which the COPY command commits changes to the database. COPY commits rows to the destination database each time it copies *n* row batches. Valid values are zero to 5000. You can set the size of a batch with the ARRAYSIZE variable. If you set COPYCOMMIT to zero, COPY performs a commit only at the end of a copy operation.

COPYTYPECHECK {OFF|ON}

Sets the suppression of the comparison of datatypes while inserting or appending to tables with the COPY command. This is to facilitate copying to DB2, which requires that a CHAR be copied to a DB2 DATE.

DEF[INE] {&|*c*|OFF|ON}

Sets the character used to prefix substitution variables to *c*. ON or OFF controls whether SQL*Plus will scan commands for substitution variables and replace them with their values. ON changes the value of *c* back to the default '&', not the most recently used character. The setting of DEFINE to OFF overrides the setting of the SCAN variable. For more information on the SCAN variable, see the [SET SCAN](#) command in [Appendix F](#).

DESCRIBE [DEPTH {1|*n*|ALL}][LINENUM {ON|OFF}][INDENT {ON|OFF}]

Sets the depth of the level to which you can recursively describe an object. The valid range of the DEPTH clause is from 1 to 50. If you SET DESCRIBE DEPTH ALL, then the depth will be set to 50, which is the maximum level allowed. You can also display the line number and indentation of the attribute or column name when an object contains multiple object types. Use the SET LINESIZE command to control the width of the data displayed.

For more information about describing objects, see the DESCRIBE command earlier in this chapter.

ECHO {OFF|ON}

Controls whether the START command lists each command in a command file as the command is executed. ON lists the commands; OFF suppresses the listing.

EDITF[ILE] *file_name***[.ext]**

Sets the default filename for the EDIT command. For more information about the EDIT command, see EDIT in this chapter.

You can include a path and/or file extension. For information on changing the default extension, see the SUFFIX variable of this command. The default filename and maximum filename length are operating system specific.

EMB[EDDED] {**OFF**|**ON**}

Controls where on a page each report begins. OFF forces each report to start at the top of a new page. ON allows a report to begin anywhere on a page. Set EMBEDDED to ON when you want a report to begin printing immediately following the end of the previously run report.

ESC[APE] {\|*c*|**OFF**|**ON**}

Defines the character you enter as the escape character. OFF undefines the escape character. ON enables the escape character. ON changes the value of *c* back to the default "\".

You can use the escape character before the substitution character (set through SET DEFINE) to indicate that SQL*Plus should treat the substitution character as an ordinary character rather than as a request for variable substitution.

FEED[BACK] {**6**|*n*|**OFF**|**ON**}

Displays the number of records returned by a query when a query selects at least *n* records. ON or OFF turns this display on or off. Turning feedback ON sets *n* to 1. Setting feedback to zero is equivalent to turning it OFF.

FLAGGER {**OFF**|**ENTRY** |**INTERMED**[**IATE**] |**FULL**}

Checks to make sure that SQL statements conform to the ANSI/ISO SQL92 standard. If any non-standard constructs are found, the Oracle Server flags them as errors and displays the violating syntax. This is the equivalent of the SQL language ALTER SESSION SET FLAGGER command.

You may execute SET FLAGGER even if you are not connected to a database. FIPS flagging will remain in effect across SQL*Plus sessions until a SET FLAGGER OFF (or ALTER SESSION SET FLAGGER = OFF) command is successful or you exit SQL*Plus.

When FIPS flagging is enabled, SQL*Plus displays a warning for the CONNECT, DISCONNECT, and ALTER SESSION SET FLAGGER commands, even if they are successful.

FLU[SH] {**OFF**|**ON**}

Controls when output is sent to the user's display device. OFF allows the host operating system to buffer output. ON disables buffering.

Use OFF only when you run a command file non-interactively (that is, when you do not need to see output and/or prompts until the command file finishes running). The use of FLUSH OFF may improve performance by reducing the amount of program I/O.

HEA[DING] {OFF|ON}

Controls printing of column headings in reports. ON prints column headings in reports; OFF suppresses column headings.

The SET HEADING OFF command will not affect the column width displayed, and only suppresses the printing of the column header itself.

HEADS[EP] {||c|OFF|ON}

Defines the character you enter as the heading separator character. The heading separator character cannot be alphanumeric or white space. You can use the heading separator character in the COLUMN command and in the old forms of BTITLE and TTITLE to divide a column heading or title onto more than one line. ON or OFF turns heading separation on or off. When heading separation is OFF, SQL*Plus prints a heading separator character like any other character. ON changes the value of *c* back to the default "|".

INSTANCE [*instance_path*|LOCAL]

Changes the default instance for your session to the specified instance path. Using the SET INSTANCE command does not connect to a database. The default instance is used for commands when no instance is specified.

Any commands preceding the first use of SET INSTANCE communicate with the default instance.

To reset the instance to the default value for your operating system, you can either enter SET INSTANCE with no *instance_path* or SET INSTANCE LOCAL. See your operating system-specific Oracle documentation for a description of how to set the initial default instance.

Note, you can only change the instance when you are not currently connected to any instance. That is, you must first make sure that you have disconnected from the current instance, then set or change the instance, and reconnect to an instance in order for the new setting to be enabled.

This command may only be issued when Net8 is running. You can use any valid Net8 connect string as the specified instance path. See your operating system-specific Oracle documentation for a complete description of how your operating system specifies Net8 connect strings. The maximum length of the instance path is 64 characters.

LIN[ESIZE] {80|*n*}

Sets the total number of characters that SQL*Plus displays on one line before beginning a new line. It also controls the position of centered and right-aligned text in TTITLE, BTITLE, REPHEADER and REPFOOTER. You can define LINESIZE as a value from 1 to a maximum that is system dependent. Refer to the Oracle installation and user's manual(s) provided for your operating system.

LOBOF[FSET] {*n*|**1**}

Sets the starting position from which CLOB and NCLOB data is retrieved and displayed.

LOGSOURCE [*pathname*]

Specifies the location from which archive logs are retrieved during recovery. The default value is set by the LOG_ARCHIVE_DEST initialization parameter. Using the SET LOGSOURCE command without a pathname restores the default location.

LONG {**80**|*n*}

Sets maximum width (in bytes) for displaying LONG, CLOB and NCLOB values; and for copying LONG values. The maximum value of *n* is 2 gigabytes.

LONGC[HUNKSIZE] {**80**|*n*}

Sets the size (in bytes) of the increments in which SQL*Plus retrieves a LONG, CLOB or NCLOB value.

NEWP[AGE] {**1**|*n*|**NONE**}

Sets the number of blank lines to be printed from the top of each page to the top title. A value of zero places a formfeed at the beginning of each page (including the first page) and clears the screen on most terminals. If you set NEWPAGE to NONE, SQL*Plus does not print a blank line or formfeed between the report pages.

NULL *text*

Sets the text that represents a null value in the result of a SQL SELECT command. Use the NULL clause of the COLUMN command to override the setting of the NULL variable for a given column.

NUMF[ORMAT] *format*

Sets the default format for displaying numbers. Enter a number format for *format*. For number format descriptions, see the FORMAT clause of the COLUMN command in this chapter.

NUM[WIDTH] {**10**|*n*}

Sets the default width for displaying numbers. For number format descriptions, see the FORMAT clause of the COLUMN command in this chapter.

PAGES[IZE] {**24**|*n*}

Sets the number of lines in each page. You can set PAGESIZE to zero to suppress all headings, page breaks, titles, the initial blank line, and other formatting information.

PAUSE [*SE*] {**OFF**|**ON**|*text*}

Allows you to control scrolling of your terminal when running reports. ON causes SQL*Plus to pause at the beginning of each page of report output. You must press [Return] after each pause. The *text* you enter specifies the text to be displayed each time SQL*Plus pauses. If you enter multiple words, you must enclose *text* in single quotes.

You can embed terminal-dependent escape sequences in the PAUSE command. These sequences allow you to create inverse video messages or other effects on terminals that support such characteristics.

RECSEP {**WRAPPED**|**EA**[*CH*]|**OFF**}
RECSEPCHAR { |*c*}

Display or print record separators. A record separator consists of a single line of the RECSEPCHAR (record separating character) repeated LINESIZE times.

RECSEPCHAR defines the record separating character. A single space is the default.

RECSEP tells SQL*Plus where to make the record separation. For example, if you set RECSEP to WRAPPED, SQL*Plus prints a record separator only after wrapped lines. If you set RECSEP to EACH, SQL*Plus prints a record separator following every row. If you set RECSEP to OFF, SQL*Plus does not print a record separator.

SERVEROUT[*PUT*] {**OFF**|**ON**} [**SIZE** *n*] [**FOR**[*MAT*]
 {**WRAPPED**|**WORD_WRAPPED**|**TRUNCATED**}]

Controls whether to display the output (that is, DBMS_OUTPUT.PUT_LINE) of stored procedures or PL/SQL blocks in SQL*Plus. OFF suppresses the output of DBMS_OUTPUT.PUT_LINE; ON displays the output.

SIZE sets the number of bytes of the output that can be buffered within the Oracle8i database server. The default for *n* is 2000. *n* cannot be less than 2000 or greater than 1,000,000.

When WRAPPED is enabled SQL*Plus wraps the server output within the line size specified by SET LINESIZE, beginning new lines when required.

When WORD_WRAPPED is enabled, each line of server output is wrapped within the line size specified by SET LINESIZE. Lines are broken on word boundaries. SQL*Plus left justifies each line, skipping all leading whitespace.

When TRUNCATED is enabled, each line of server output is truncated to the line size specified by SET LINESIZE.

For each FORMAT, every server output line begins on a new output line.

For more information on DBMS_OUTPUT.PUT_LINE, see your *Oracle8i Supplied Packages Reference*.

SHIFT[INOUT] {VIS[IBLE] | INV[ISIBLE]}

Allows correct alignment for terminals that display shift characters. The SET SHIF TINOUT command is useful for terminals which display shift characters together with data (for example, IBM 3270 terminals). You can only use this command with shift sensitive character sets (for example, JA16DBCS).

Use VISIBLE for terminals that display shift characters as a visible character (for example, a space or a colon). INVISIBLE is the opposite and does not display any shift characters.

SHOW[MODE] {OFF | ON}

Controls whether SQL*Plus lists the *old* and *new* settings of a SQL*Plus system variable when you change the setting with SET. ON lists the settings; OFF suppresses the listing. SHOWMODE ON has the same behavior as the obsolete SHOWMODE BOTH.

SQLBL[ANKLINES] {ON | OFF}

Controls whether SQL*Plus allows blank lines within a SQL command. ON interprets blank lines and new lines as part of a SQL command. OFF, the default value, does not allow blank lines or new lines in a SQL command. SQL*Plus returns to the default behavior when a SQLTERMINATOR or BLOCKTERMINATOR is encountered.

SQLC[ASE] {MIX[ED] | LO[WER] | UP[PER]}

Converts the case of SQL commands and PL/SQL blocks just prior to execution. SQL*Plus converts all text within the command, including quoted literals and identifiers, as follows:

- uppercase if SQLCASE equals UPPER
- lowercase if SQLCASE equals LOWER
- unchanged if SQLCASE equals MIXED

SQLCASE does not change the SQL buffer itself.

SQLCO[NTINUE] {> | text}

Sets the character sequence SQL*Plus displays as a prompt after you continue a SQL*Plus command on an additional line using a hyphen (-).

SQLN[UMBER] {OFF | ON}

Sets the prompt for the second and subsequent lines of a SQL command or PL/SQL block. ON sets the prompt to be the line number. OFF sets the prompt to the value of SQLPROMPT.

SQLPRE[*FIX*] {#|*c*}

Sets the SQL*Plus prefix character. While you are entering a SQL command or PL/SQL block, you can enter a SQL*Plus command on a separate line, prefixed by the SQL*Plus prefix character. SQL*Plus will execute the command immediately without affecting the SQL command or PL/SQL block that you are entering. The prefix character must be a non-alphanumeric character.

SQLP[*ROMPT*] {SQL>|*text*}

Sets the SQL*Plus command prompt.

SQLT[*ERMINATOR*] {;|*c*|OFF|ON}

Sets the character used to end and execute SQL commands to *c*. OFF means that SQL*Plus recognizes no command terminator; you terminate a SQL command by entering an empty line. ON resets the terminator to the default semicolon (;).

SUF[*FIX*] {SQL|*text*}

Sets the default file extension that SQL*Plus uses in commands that refer to command files. SUFFIX does not control extensions for spool files.

TAB {OFF|ON}

Determines how SQL*Plus formats white space in terminal output. OFF uses spaces to format white space in the output. ON uses the TAB character. TAB settings are every eight characters. The default value for TAB is system dependent.

TERM[*OUT*] {OFF|ON}

Controls the display of output generated by commands executed from a command file. OFF suppresses the display so that you can spool output from a command file without seeing the output on the screen. ON displays the output. TERMOUT OFF does not affect output from commands you enter interactively.

TI[*ME*] {OFF|ON}

Controls the display of the current time. ON displays the current time before each command prompt. OFF suppresses the time display.

TIMI[*NG*] {OFF|ON}

Controls the display of timing statistics. ON displays timing statistics on each SQL command or PL/SQL block run. OFF suppresses timing of each command. For information about the data SET TIMING ON displays, see the Oracle installation and user's manual(s) provided for your operating system.

Refer to the TIMING command for information on timing multiple commands.

TRIM[OUT] {OFF|ON}

Determines whether SQL*Plus allows trailing blanks at the end of each displayed line. ON removes blanks at the end of each line, improving performance especially when you access SQL*Plus from a slow communications device. OFF allows SQL*Plus to display trailing blanks. TRIMOUT ON does not affect spooled output.

TRIMS[POOL] {ON|OFF}

Determines whether SQL*Plus allows trailing blanks at the end of each spooled line. ON removes blanks at the end of each line. OFF allows SQL*Plus to include trailing blanks. TRIMSPool ON does not affect terminal output.

UND[ERLINE] {-|c|ON|OFF}

Sets the character used to underline column headings in SQL*Plus reports to *c*. Note, *c* cannot be an alphanumeric character or a white space. ON or OFF turns underlining on or off. ON changes the value of *c* back to the default "-".

VER[IFY] {OFF|ON}

Controls whether SQL*Plus lists the text of a SQL statement or PL/SQL command before and after SQL*Plus replaces substitution variables with values. ON lists the text; OFF suppresses the listing.

WRA[P] {OFF|ON}

Controls whether SQL*Plus truncates the display of a SELECTed row if it is too long for the current line width. OFF truncates the SELECTed row; ON allows the SELECTed row to wrap to the next line.

Use the WRAPPED and TRUNCATED clauses of the COLUMN command to override the setting of WRAP for specific columns.

Usage Notes

SQL*Plus maintains system variables (also called SET command variables) to allow you to establish a particular environment for a SQL*Plus session. You can change these system variables with the SET command and list them with the SHOW command.

SET ROLE and SET TRANSACTION are SQL commands (see the *Oracle8i SQL Reference* for more information). When not followed by the keywords TRANSACTION or ROLE, SET is assumed to be a SQL*Plus command.

Examples

The following examples show sample uses of selected SET command variables.

APPINFO

To display the setting of APPINFO, enter

```
SQL> SHOW APPINFO
SQL> appinfo is ON and set to "SQL*Plus"
```

To change the default text, enter

```
SQL> SET APPI 'This is SQL*Plus'
SQL> SHOW APPINFO
SQL> appinfo is ON and set to "This is SQL*Plus"
```

To make sure that registration has taken place, enter

```
SQL> VARIABLE MOD VARCHAR2(50)
SQL> VARIABLE ACT VARCHAR2(40)
SQL> EXECUTE DBMS_APPLICATION_INFO.READ_MODULE(:MOD, :ACT);
SQL> PRINT MOD
MOD
-----
This is SQL*Plus
```

AUTORECOVERY

To set the recovery mode to AUTOMATIC, enter

```
SQL> SET AUTORECOVERY ON
SQL> RECOVER DATABASE
```

CMDSEP

To specify a TTITLE and format a column on the same line, enter

```
SQL> SET CMDSEP +
SQL> TTITLE LEFT 'SALARIES' + COLUMN SAL FORMAT $9,999
SQL> SELECT ENAME, SAL FROM EMP
      2  WHERE JOB = 'CLERK';
```

The following output results:

SALARIES	
ENAME	SAL
SMITH	\$800
ADAMS	\$1,100
JAMES	\$950
MILLER	\$1,300

COLSEP

To set the column separator to "|" enter

```
SQL> SET COLSEP '|'
SQL> SELECT ENAME, JOB, DEPTNO
```

```

2 FROM EMP
3 WHERE DEPTNO = 20;

```

The following output results:

ENAME	JOB	DEPTNO
SMITH	CLERK	20
JONES	MANAGER	20
SCOTT	ANALYST	20
ADAMS	CLERK	20
FORD	ANALYST	20

COMPATIBILITY

To run a command file, SALARY.SQL, created with Oracle7, enter

```

SQL> SET COMPATIBILITY V7
SQL> START SALARY

```

After running the file, reset compatibility to V8 to run command files created with Oracle8:

```

SQL> SET COMPATIBILITY V8

```

Alternatively, you can add the command SET COMPATIBILITY V7 to the beginning of the command file, and reset COMPATIBILITY to V8 at the end of the file.

DESCRIBE

To describe the object emp_object to a depth of two levels, and indent the output while also displaying line numbers, first describe the object as follows:

```

SQL> desc emp_object

```

The following output results:

Name	Null	Type
EMPLOYEE		RECUR_PERSON
NAME		VARCHAR2(20)
ADDR		RECUR_ADDRESS
ADDR1		RECUR_ADDRESS1
DOB		DATE
GENDER		VARCHAR2(10)
DEPT		RECUR_DEPARTMENT
DEPTNO		NUMBER
DEPT_NAME		VARCHAR2(20)
LOCATION		VARCHAR2(20)
START_DATE		DATE
POSITION		VARCHAR2(1)
SAL		RECUR_SALARY
ANNUAL_SAL		NUMBER(10,2)
EMP_TYPE		VARCHAR2(1)
COMM		NUMBER(10,2)

PENALTY_RATE

NUMBER(5,2)

To format emp_object so that the output displays with indentation and line numbers, use the SET DESCRIBE command as follows:

```
SQL> SET DESCRIBE DEPTH 2 LINENUM ON INDENT ON
```

To display the above settings, enter

```
SQL> DESCRIBE emp_object
```

The following output results:

	Name	Null	Type
-----	-----	-----	-----
1	EMPLOYEE		RECUR_PERSON
2	1 NAME		VARCHAR2(20)
3	1 ADDR		RECUR_ADDRESS
4	1 ADDR1		RECUR_ADDRESS1
5	1 DOB		DATE
6	1 GENDER		VARCHAR2(10)
7	DEPT		RECUR_DEPARTMENT
8	7 DEPTNO		NUMBER
9	7 DEPT_NAME		VARCHAR2(20)
10	7 LOCATION		VARCHAR2(20)
11	START_DATE		DATE
12	POSITION		VARCHAR2(1)
13	SAL		RECUR_SALARY
14	13 ANNUAL_SAL		NUMBER(10,2)
15	13 EMP_TYPE		VARCHAR2(1)
16	13 COMM		NUMBER(10,2)
17	13 PENALTY_RATE		NUMBER(5,2)

ESCAPE

If you define the escape character as an exclamation point (!), then

```
SQL> SET ESCAPE !
SQL> ACCEPT v1 PROMPT 'Enter !&1:'
```

displays this prompt:

```
Enter &1:
```

HEADING

To suppress the display of column headings in a report, enter

```
SQL> SET HEADING OFF
```

If you then run a SQL SELECT command,

```
SQL> SELECT ENAME, SAL FROM EMP  
2 WHERE JOB = 'CLERK';
```

the following output results:

ADAMS	1100
JAMES	950
MILLER	1300

INSTANCE

To set the default instance to "PROD1" enter

```
SQL> SET INSTANCE PROD1
```

To set the instance back to the default or local, enter

```
SQL> SET INSTANCE local
```

LOBOFFSET

To set the starting position from which a CLOB column's data is retrieved to the 22nd position, enter

```
SQL> SET LOBOFFSET 22
```

The CLOB data will wrap on your screen; SQL*Plus will not truncate until the 23rd character.

LOGSOURCE

To set the default location of log files for recovery to the directory "/usr/oracle81/dbs/arch" enter

```
SQL> SET LOGSOURCE "/usr/oracle81/dbs/arch"  
SQL> RECOVER DATABASE
```

LONG

To set the maximum width for displaying and copying LONG values to 500, enter

```
SQL> SET LONG 500
```

The LONG data will wrap on your screen; SQL*Plus will not truncate until the 501st character.

LONGCHUNKSIZE

To set the size of the increments in which SQL*Plus retrieves LONG values to 100 characters, enter

```
SQL> SET LONGCHUNKSIZE 100
```

The LONG data will be retrieved in increments of 100 characters until the entire value is retrieved or the value of SET LONG is reached.

SERVEROUTPUT

To enable the display of DBMS_OUTPUT.PUT_LINE, enter

```
SQL> SET SERVEROUTPUT ON
```

The following example shows what happens when you execute an anonymous procedure with SET SERVEROUTPUT ON:

```
SQL> BEGIN
  2  DBMS_OUTPUT.PUT_LINE('Task is complete');
  3  END;
  4  /
Task is complete.
```

PL/SQL procedure successfully completed.

The following example shows what happens when you create a trigger with SET SERVEROUTPUT ON:

```
SQL> CREATE TRIGGER SERVER_TRIG BEFORE INSERT OR UPDATE -
> OR DELETE
  2  ON SERVER_TAB
  3  BEGIN
  4  DBMS_OUTPUT.PUT_LINE('Task is complete. ');
  5  END;
  6  /
Trigger created.
SQL> INSERT INTO SERVER_TAB VALUES ('TEXT');
Task is complete.
1 row created.
```

To set the output to WORD_WRAPPED, enter

```
SQL> SET SERVEROUTPUT ON FORMAT WORD_WRAPPED
SQL> SET LINESIZE 20
SQL> BEGIN
  2  DBMS_OUTPUT.PUT_LINE('If there is nothing left to do');
  3  DBMS_OUTPUT.PUT_LINE('shall we continue with plan B?');
  4  end;
  5  /
If there is nothing
left to do
shall we continue
with plan B?
```

To set the output to TRUNCATED, enter

```
SQL> SET SERVEROUTPUT ON FORMAT TRUNCATED
```

```

SQL> SET LINESIZE 20
SQL> BEGIN
  2  DBMS_OUTPUT.PUT_LINE('If there is nothing left to do');
  3  DBMS_OUTPUT.PUT_LINE('shall we continue with plan B?');
  4  END;
  5  /
If there is nothing
shall we continue wi

```

SHIFTINOUT

To enable the display of shift characters, enter

```

SQL> SET SHIFTINOUT VISIBLE
SQL> SELECT ENAME, JOB FROM EMP;

```

The following output results:

ENAME	JOB
:JJOO:	:AABBCC:
:AA:abc	:DDEE:e

where ":" = shift character

uppercase = multibyte character

lowercase = singlebyte character

Note:

This example illustrates that the columns are aligned correctly. The data used in this example is an illustration only and does not represent real data.

SQLBLANKLINES

To preserve blank lines in a SQL statement, enter

```

SQL> SET SQLBLANKLINES ON
SQL> SELECT *
  2
  3 FROM
  4
  5 DUAL
  6
  7 ;

```

The following output results:

D
-
X

SQLCONTINUE

To set the SQL*Plus command continuation prompt to an exclamation point followed by a space, enter

```
SQL> SET SQLCONTINUE ' ! '
```

SQL*Plus will prompt for continuation as follows:

```
SQL> TTITLE 'YEARLY INCOME' -  
! RIGHT SQL.PNO SKIP 2 -  
! CENTER 'PC DIVISION'  
SQL>
```

SUFFIX

To set the default command-file extension to UFI, enter

```
SQL> SET SUFFIX UFI
```

If you then enter

```
SQL> GET EXAMPLE
```

SQL*Plus will look for a file named EXAMPLE with an extension of UFI instead of EXAMPLE with an extension of SQL.

SHOW

Purpose

Shows the value of a SQL*Plus system variable or the current SQL*Plus environment.

Syntax

```
SHO[W] option
```

where *option* represents one of the following terms or clauses:

```
system_variable  
ALL  
BTI[TLE]  
ERR[ORS] [ {FUNCTION|PROCEDURE|PACKAGE|PACKAGE BODY|  
TRIGGER|VIEW|TYPE|TYPE BODY} [schema.]name ]
```

LNO
 PARAMETERS [*parameter_name*]
 PNO
 REL[EASE]
 REPF[OOTER]
 REPH[EADER]
 SGA
 SPOOL[L]
 SQLCODE
 TTI[TLE]
 USER

Terms and Clauses

Refer to the following list for a description of each term or clause:

system_variable

Represents any system variable set by the SET command.

ALL

Lists the settings of all SHOW options, except ERRORS, in alphabetical order.

BTI[TLE]

Shows the current BTITLE definition.

**ERR[ORS] [{FUNCTION|PROCEDURE|PACKAGE|PACKAGE BODY
 |TRIGGER|VIEW|TYPE|TYPE BODY} [*schema.*]*name*]**

Shows the compilation errors of a stored procedure (includes stored functions, procedures, and packages). After you use the CREATE command to create a stored procedure, a message is displayed if the stored procedure has any compilation errors. To see the errors, you use SHOW ERRORS.

When you specify SHOW ERRORS with no arguments, SQL*Plus shows compilation errors for the most recently created or altered stored procedure. When you specify the type (function, procedure, package, package body, trigger, view, type, or type body) and the name of the PL/SQL stored procedure, SQL*Plus shows errors for that stored procedure. For more information on compilation errors, see your *PL/SQL User's Guide and Reference*.

schema contains the named object. If you omit *schema*, SHOW ERRORS assumes the object is located in your current schema.

SHOW ERRORS output displays the line and column number of the error (LINE/COL) as well as the error itself (ERROR). LINE/COL and ERROR have default widths of 8 and 65, respectively. You can alter these widths using the COLUMN command.

LNO

Shows the current line number (the position in the current page of the display and/or spooled output).

PARAMETERS [*parameter_name*]

Displays the current values for one or more initialization parameters. You can use a string after the command to see a subset of parameters whose names include that string. For example, if you enter:

```
SQL> SHOW PARAMETERS COUNT
```

you would see:

NAME	TYPE	VALUE
-----	-----	-----
db_file_multiblock_read_count	integer	12
spin_count	integer	0

The SHOW PARAMETERS command, without any string following the command, displays all initialization parameters.

Note, your output may vary depending on the version and configuration of the Oracle database server to which you are connected.

PNO

Shows the current page number.

REL[EASE]

Shows the release number of Oracle that SQL*Plus is accessing.

REPF[OOTER]

Shows the current REPFOOTER definition.

REPH[EADER]

Shows the current REPHEADER definition.

SPOO[L]

Shows whether output is being spooled.

SGA

Displays information about the current instance's System Global Area.

SQLCODE

Shows the value of SQL.SQLCODE (the SQL return code of the most recent operation).

TTI[TITLE]

Shows the current TTITLE definition.

USER

Shows the username under which you are currently accessing SQL*Plus.

Examples

To list the current LINESIZE, enter

```
SQL> SHOW LINESIZE
```

If the current linesize equals 80 characters, SQL*Plus will give the following response:

```
linesize 80
```

The following example illustrates how to create a stored procedure and then show its compilation errors:

```
SQL> connect system/manager
SQL> create procedure scott.procl as
SQL> begin
SQL>   :pl := 1;
SQL> end;
SQL> /
Warning: Procedure created with compilation errors.
SQL> show errors
Errors for PROCEDURE SCOTT.PROC1:
LINE/COL ERROR
-----
3/3      PLS-00049: bad bind variable 'P1'
SQL> show errors procedure procl
No errors.
SQL> show errors procedure scott.procl
Errors for PROCEDURE SCOTT.PROC1:
LINE/COL ERROR
-----
3/3      PLS-00049: bad bind variable 'P1'
```

To show whether AUTORECOVERY is enabled, enter

```
SQL> SHOW AUTORECOVERY
autorecovery ON
```

To display the connect string for the default instance, enter

```
SQL> SHOW INSTANCE
instance "local"
```

To display the location for archive logs, enter

```
SQL> SHOW LOGSOURCE
logsource "/usr/oracle81/dbs/arch"
```

To display information about the SGA, enter

```
SQL> SHOW SGA
```

Total System Global Area	7629732 bytes
Fixed Size	60324 bytes
Variable Size	6627328 bytes
Database Buffers	409600 bytes
Redo Buffers	532480 bytes

SHUTDOWN

Purpose

Shuts down a currently running Oracle instance, optionally closing and dismounting a database.

Syntax

SHUTDOWN [ABORT|IMMEDIATE|NORMAL|TRANSACTIONAL]

Terms and Clauses

Refer to the following list for a description of each term or clause:

ABORT

Proceeds with the fastest possible shutdown. Does not wait for calls to complete or users to disconnect. Does not close or dismount the database, but does shut down the instance. Requires instance recovery on next startup. You must use this option if a background process terminates abnormally.

IMMEDIATE

Does not wait for current calls to complete, prohibits further connects, and closes and dismounts the database. Finally, shuts down the instance. Does not wait for connected users to disconnect. Does not require instance recovery on next startup.

NORMAL

Waits for currently connected users to disconnect from the database, prohibits further connects, and closes and dismounts the database. Finally, shuts down the instance. Does not require instance recovery on next startup. NORMAL is the default option.

TRANSACTIONAL

Performs shutdown of an instance while minimizing interruption to clients. No client can start a new transaction on the instance. If a client attempts to start a new transaction, they are disconnected. After all transactions have either been committed or aborted, any client still connected to the instance is disconnected. At this point, the instance shuts down just as it would when a SHUTDOWN IMMEDIATE is submitted.

Using SHUTDOWN TRANSACTIONAL prevents clients from losing work, and at the same time, does not require all users to log off.

Usage Notes

SHUTDOWN with no arguments is equivalent to SHUTDOWN NORMAL.

You must be connected to a database as SYSOPER, or SYSDBA. You cannot connect via a multi-threaded server. For more information about connecting to a database, see the [CONNECT](#) command earlier in this chapter.

Example

To shutdown the database in normal mode, enter

```
SQL> SHUTDOWN
Database closed.
Database dismounted.
Oracle instance shut down.
```

SPOOL

Purpose

Stores query results in an operating system file and, optionally, sends the file to a printer.

Syntax

```
SPO[OL] [file_name [.ext] | OFF | OUT]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

***file_name* [.ext]**

Represents the name of the file to which you wish to spool. SPOOL followed by *file_name* begins spooling displayed output to the named file. If you do not specify an extension, SPOOL uses a default extension (LST or LIS on most systems).

OFF

Stops spooling.

OUT

Stops spooling and sends the file to your host computer's standard (default) printer.

Enter SPOOL with no clauses to list the current spooling status.

Usage Notes

To spool output generated by commands in a command file without displaying the output on the screen, use SET TERMOUT OFF. SET TERMOUT OFF does not affect output from commands run interactively.

Examples

To record your displayed output in a file named DIARY using the default file extension, enter

```
SQL> SPOOL DIARY
```

To stop spooling and print the file on your default printer, enter

```
SQL> SPOOL OUT
```

START

Purpose

Executes the contents of the specified command file.

Syntax

```
STA[RT] file_name[.ext] [arg ...]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

***file_name*[.ext]**

Represents the command file you wish to execute. The file can contain any command that you can run interactively.

If you do not specify an extension, SQL*Plus assumes the default command-file extension (normally SQL). For information on changing this default extension, see the SUFFIX variable of the SET command in this chapter.

When you enter `START file_name.ext`, SQL*Plus searches for a file with the filename and extension you specify in the current default directory. If SQL*Plus does not find such a file, SQL*Plus will search a system-dependent path to find the file. Some operating systems may not support the path search. Consult the Oracle installation and user's manual(s) provided for your operating system for specific information related to your operating system environment.

***arg* ...**

Represent data items you wish to pass to parameters in the command file. If you enter one or more arguments, SQL*Plus substitutes the values into the parameters (&1, &2, and so forth) in the command file. The first argument replaces each occurrence of &1, the second replaces each occurrence of &2, and so forth.

The START command **DEFINES** the parameters with the values of the arguments; if you START the command file again in this session, you can enter new arguments or omit the arguments to use the old values.

For more information on using parameters, refer to the subsection "[Passing Parameters through the START Command](#)" under "[Writing Interactive Commands](#)" in [Chapter 3](#).

Usage Notes

The @ ("at" sign) and @@ (double "at" sign) commands function similarly to START. Disabling the START command in the Product User Profile also disables the @ and @@ commands. See the [@](#) and [@@](#) commands in this chapter for further information on these commands.

The EXIT or QUIT commands in a command file terminate SQL*Plus.

Example

A file named PROMOTE with the extension SQL, used to promote employees, might contain the following command:

```
SELECT * FROM EMP
WHERE MGR=&1 AND JOB='&2' AND SAL>&3;
```

To run this command file, enter

```
SQL> START PROMOTE 7280 CLERK 950
```

SQL*Plus then executes the following command:

```
SELECT * FROM EMP
WHERE MGR=7280 AND JOB='CLERK' AND SAL>950;
```

STARTUP

Purpose

Starts an Oracle instance with several options, including mounting and opening a database.

Syntax

```
STARTUP [FORCE][RESTRICT][PFILE=filename][MOUNT[OPEN[RECOVER]]
[database]][mount_options][NOMOUNT]
```

where *mount_options* requires the following syntax:

```
[EXCLUSIVE|[PARALLEL|SHARED]][RETRY]]
```

Terms and Clauses

Refer to the following list for a description of each term and clause:

FORCE

Shuts down the current Oracle instance (if it is running) with SHUTDOWN mode ABORT, before restarting it. If the current instance is running and FORCE is not specified, an error results. FORCE is useful while debugging and under abnormal circumstances. It should not normally be used.

RESTRICT

Only allows Oracle users with the RESTRICTED SESSION system privilege to connect to the database. Later, you can use the ALTER SYSTEM command to disable the restricted session feature.

PFILE=filename

Causes the specified parameter file to be used while starting up the instance.

MOUNT

Mounts a database but does not open it.

OPEN

Mounts and opens the specified database.

NOMOUNT

Causes the database not to be mounted upon instance startup. Cannot be used with SHARED, EXCLUSIVE, PARALLEL, MOUNT, or OPEN.

RECOVER

Specifies that media recovery should be performed, if necessary, before starting the instance. STARTUP RECOVER has the same effect as issuing the RECOVER DATABASE command and starting an instance. Only complete recovery is possible with the RECOVER option.

Recovery proceeds, if necessary, as if AUTORECOVERY is set to ON, regardless of whether or not AUTORECOVERY is enabled. If a redo log file is not found in the expected location, recovery continues as if AUTORECOVERY is disabled, by prompting you with the suggested location and name of the subsequent log files that need to be applied.

If recovery fails using the RECOVER option, the database remains mounted and closed.

database

The name of the database to mount or open. If no database name is specified, the database name is taken from the initialization parameter DB_NAME.

EXCLUSIVE

Signifies that the database can only be mounted and opened by the current instance (it cannot be opened simultaneously by multiple instances). Cannot be used with

SHARED, PARALLEL, or NOMOUNT. If no mounting option is specified, EXCLUSIVE is assigned by default.

PARALLEL

Must be specified if the database is to be mounted by multiple instances concurrently. Cannot be used with EXCLUSIVE or NOMOUNT. Invalid if the initialization parameter SINGLE_PROCESS is set to TRUE.

SHARED

Synonym for PARALLEL.

RETRY

Specifies that opening the database should be attempted every five seconds if the instance is busy being recovered by another instance. When an instance is being recovered by another instance, the down instance cannot open the database until recovery is complete. If the database cannot be opened for any other reason, RETRY does not attempt to open the database again. This option is only available for instances operating in PARALLEL mode.

Usage Notes

You must be connected to a database as SYSOPER, or SYSDBA. You cannot be connected via a multi-threaded server.

Examples

To start an instance using the standard parameter file, mount the default database in exclusive mode, and open the database, enter

```
SQL> STARTUP
```

or enter

```
SQL> STARTUP OPEN database EXCLUSIVE
```

To start an instance using the standard parameter file, mount the default database in parallel mode, and open the database, enter

```
SQL> STARTUP PARALLEL  
SQL> STARTUP OPEN database PARALLEL
```

To restart an instance that went down in parallel mode and may not yet have been recovered by other instances, use the RETRY option:

```
SQL> STARTUP PARALLEL RETRY
```

To shutdown the current instance, immediately restart it without mounting or opening, and allow only database administrators to connect, enter

```
SQL> STARTUP FORCE NOMOUNT RESTRICT
```

To start an instance using the parameter file TESTPARM without mounting the database, enter

```
SQL> STARTUP PFILE=testparm NOMOUNT
```

To shutdown a particular database, immediately restart and open it in parallel mode, allow access only to database administrators, and use the parameter file MYINIT.ORA. enter

```
SQL> STARTUP OPEN database PFILE=myinit.ora FORCE SHARED RESTRICT
```

To startup an instance and mount but not open a database, enter

```
SQL> CONNECT INTERNAL
Connected to an idle instance.
SQL> STARTUP MOUNT
ORACLE instance started.
```

Total System Global Area	7629732 bytes
Fixed Size	60324 bytes
Variable Size	6627328 bytes
Database Buffers	409600 bytes
Redo Buffers	532480 bytes

STORE

Purpose

Saves attributes of the current SQL*Plus environment in a host operating system file (a command file).

Syntax

```
STORE {SET} file_name[.ext] [CRE[ATE]|REP[LACE]|APP[END]]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

SET

Saves the values of the system variables.

Refer to the SAVE command for information on the other terms and clauses in the STORE command syntax.

Usage Notes

This command creates a command file which can be executed with the START, @ or @@ commands.

If you want to store a file under a name identical to a STORE command clause (that is, CREATE, REPLACE or APPEND), you must put the name in single quotes or specify a file extension.

Example

To store the current SQL*Plus system variables in a file named DEFAULTENV with the default command-file extension, enter

```
SQL> STORE SET DEFAULTENV
```

To append the current SQL*Plus system variables to an existing file called DEFAULTENV with the extension OLD, enter

```
SQL> STORE SET DEFAULTENV.OLD APPEND
```

TIMING

Purpose

Records timing data for an elapsed period of time, lists the current timer's name and timing data, or lists the number of active timers.

Syntax

```
TIMI[NG] [START text | SHOW | STOP]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

START *text*

Sets up a timer and makes *text* the name of the timer. You can have more than one active timer by STARTing additional timers before STOPping the first; SQL*Plus nests each new timer within the preceding one. The timer most recently STARTed becomes the current timer.

SHOW

Lists the current timer's name and timing data.

STOP

Lists the current timer's name and timing data, then deletes the timer. If any other timers are active, the next most recently STARTed timer becomes the current timer.

Enter TIMING with no clauses to list the number of active timers.

Usage Notes

You can use this data to do a performance analysis on any commands or blocks run during the period.

For information about the data TIMING displays, see the Oracle installation and user's manual(s) provided for your operating system. Refer to [SET TIMING ON](#) for information on automatically displaying timing data after each SQL command or PL/SQL block you run.

To delete all timers, use the CLEAR TIMING command.

Examples

To create a timer named SQL_TIMER, enter

```
SQL> TIMING START SQL_TIMER
```

To list the current timer's title and accumulated time, enter

```
SQL> TIMING SHOW
```

To list the current timer's title and accumulated time and to remove the timer, enter

```
SQL> TIMING STOP
```

TTITLE

Purpose

Places and formats a specified title at the top of each report page or lists the current TTITLE definition. The old form of TTITLE is used if only a single word or string in quotes follows the TTITLE command.

For a description of the old form of TTITLE, see [TTITLE](#) in [Appendix F](#).

Syntax

```
TTI[TLE] [printspec [text|variable] ...] | [OFF|ON]
```

where *printspec* represents one or more of the following clauses used to place and format the *text*:

```
COL n  
S[KIP] [n]  
TAB n  
LE[FT]  
CE[NTER]  
R[IGHT]  
BOLD  
FORMAT text
```

Terms and Clauses

Refer to the following list for a description of each term or clause. These terms and clauses also apply to the BTITLE command.

text

Represents the title text. Enter *text* in single quotes if you want to place more than one word on a single line.

variable

Represents a user variable or any of the following system-maintained values:

- SQL.LNO (current line number)
- SQL.PNO (current page number)
- SQL.RELEASE (current Oracle release number)
- SQL.SQLCODE (current error code)
- SQL.USER (current username)

To print one of these values, reference the appropriate variable in the title. You can format *variable* with the FORMAT clause.

OFF

Turns the title off (suppresses its display) without affecting its definition.

ON

Turns the title on (restores its display). When you define a top title, SQL*Plus automatically sets TTITLE to ON.

COL *n*

Indents to column *n* of the current line (backward if column *n* has been passed). "Column" in this context means print position, not table column.

S[KIP] [*n*]

Skips to the start of a new line *n* times; if you omit *n*, one time; if you enter zero for *n*, backward to the start of the current line.

TAB *n*

Skips forward *n* columns (backward if you enter a negative value for *n*). "Column" in this context means print position, not table column.

LE[FT] | CE[NTER] | R[IGHT]

Left-align, center, and right-align data on the current line respectively. SQL*Plus aligns following data items as a group, up to the end of the *printspec* or the next LEFT, CENTER, RIGHT, or COL command. CENTER

and RIGHT use the SET LINESIZE value to calculate the position of the data item that follows.

BOLD

Prints data in bold print. SQL*Plus represents bold print on your terminal by repeating the data on three consecutive lines. On some operating systems, SQL*Plus may instruct your printer to print bolded text on three consecutive lines, instead of bold.

FORMAT *text*

Specifies a format model that determines the format of following data items, up to the next FORMAT clause or the end of the command. The format model must be a *text* constant such as A10 or \$999. See the COLUMN FORMAT command for more information on formatting and valid format models.

If the datatype of the format model does not match the datatype of a given data item, the FORMAT clause has no effect on that item.

If no appropriate FORMAT model precedes a given data item, SQL*Plus prints NUMBER values according to the format specified by SET NUMFORMAT or, if you have not used SET NUMFORMAT, the default format. SQL*Plus prints DATE values according to the default format.

Refer to the FORMAT clause of the COLUMN command in this chapter for more information on default formats.

Enter TTITLE with no clauses to list the current TTITLE definition.

Usage Notes

If you do not enter a *printspec* clause before the first occurrence of *text*, TTITLE left justifies the text. SQL*Plus interprets TTITLE in the new form if a valid *printspec* clause (LEFT, SKIP, COL, and so on) immediately follows the command name.

See COLUMN NEW_VALUE for information on printing column and DATE values in the top title.

You can use any number of constants and variables in a *printspec*. SQL*Plus displays the constants and variables in the order you specify them, positioning and formatting each constant or variable as specified by the *printspec* clauses that precede it.

The length of the title you specify with TTITLE cannot exceed 2400 characters.

The continuation character (a hyphen) will not be recognized inside a single-quoted title text string. To be recognized, the continuation character must appear outside the quotes, as follows:

```
SQL> TTITLE CENTER 'Summary Report for' -  
> 'the Month of May'
```

Examples

To define "Monthly Analysis" as the top title and to left-align it, to center the date, to right-align the page number with a three-digit format, and to display "Data in Thousands" in the center of the next line, enter

```
SQL> TTITLE LEFT 'Monthly Analysis' CENTER '23 Nov 98' -  
> RIGHT 'Page:' FORMAT 999 SQL.PNO SKIP CENTER -  
> 'Data in Thousands'
```

The following title results:

```
Monthly Analysis                23 Nov 98                Page:    1  
                                Data in Thousands
```

To suppress the top title display without changing its definition, enter

```
SQL> TTITLE OFF
```

UNDEFINE

Purpose

Deletes one or more user variables that you defined either explicitly (with the DEFINE command) or implicitly (with an argument to the START command).

Syntax

```
UNDEF[INE] variable ...
```

Terms and Clauses

Refer to the following for a description of the term or clause:

variable

Represents the name of the user variable you wish to delete. One or more user variables may be deleted in the same command.

Examples

To undefine a user variable named POS, enter

```
SQL> UNDEFINE POS
```

To undefine two user variables named MYVAR1 and MYVAR2, enter

```
SQL> UNDEFINE MYVAR1 MYVAR2
```

VARIABLE

Purpose

Declares a bind variable that can then be referenced in PL/SQL. For more information on bind variables, see ["Using Bind Variables"](#) in [Chapter 3](#). For more information about PL/SQL, see your *PL/SQL User's Guide and Reference*.

VARIABLE without arguments displays a list of all the variables declared in the session. VARIABLE followed only by a variable name lists that variable.

Syntax

```
VAR[iable] [variable [NUMBER|CHAR|CHAR (n)|NCHAR|NCHAR (n)
|VARCHAR2 (n)|NVARCHAR2 (n)|CLOB|NCLOB|REFCURSOR]]
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

variable

Represents the name of the bind variable you wish to create.

NUMBER

Creates a variable of type NUMBER with a fixed length.

CHAR

Creates a variable of type CHAR (character) with a length of one.

CHAR (*n*)

Creates a variable of type CHAR with a maximum length of *n*, up to 2000.

NCHAR

Creates a variable of type NCHAR (national character) with a length of one.

NCHAR (*n*)

Creates a variable of type NCHAR with a maximum length of *n*, up to 2000.

VARCHAR2 (*n*)

Creates a variable of type VARCHAR2 with a maximum length of *n*, up to 4000.

NVARCHAR2 (*n*)

Creates a variable of type NVARCHAR2 (NCHAR VARYING) with a maximum length of *n*, up to 4000.

CLOB

Creates a variable of type CLOB.

NCLOB

Creates a variable of type NCLOB.

REFCURSOR

Creates a variable of type REF CURSOR.

Usage Notes

Bind variables may be used as parameters to stored procedures, or may be directly referenced in anonymous PL/SQL blocks.

To display the value of a bind variable created with VARIABLE, use the PRINT command. For more information, see the PRINT command in this chapter.

To automatically display the value of a bind variable created with VARIABLE, use the SET AUTOPRINT command. For more information, see the SET AUTOPRINT command in this chapter.

Bind variables cannot be used in the COPY command or SQL statements, except in PL/SQL blocks. Instead, use substitution variables.

When you execute a VARIABLE ... CLOB or NCLOB command, SQL*Plus associates a LOB locator with the bind variable. The LOB locator is automatically populated when you execute a SELECT clob_column INTO :cv statement in a PL/SQL block. SQL*Plus closes the LOB locator after completing a PRINT statement for that bind variable, or when you exit SQL*Plus.

SQL*Plus SET commands such as SET LONG and SET LONGCHUNKSIZE and SET LOBOFFSET may be used to control the size of the buffer while PRINTing CLOB or NCLOB bind variables.

SQL*Plus REFCURSOR bind variables may be used to reference PL/SQL 2.3 or higher Cursor Variables, allowing PL/SQL output to be formatted by SQL*Plus. For more information on PL/SQL Cursor Variables, see your *PL/SQL User's Guide and Reference*.

When you execute a VARIABLE ... REFCURSOR command, SQL*Plus creates a cursor bind variable. The cursor is automatically opened by an OPEN ... FOR SELECT statement referencing the bind variable in a PL/SQL block. SQL*Plus closes the cursor after completing a PRINT statement for that bind variable, or on exit.

SQL*Plus formatting commands such as BREAK, COLUMN, COMPUTE and SET may be used to format the output from PRINTing a REFCURSOR.

A REFCURSOR bind variable may not be PRINTed more than once without re-executing the PL/SQL OPEN ... FOR statement.

Examples

The following example illustrates creating a bind variable and then setting it to the value returned by a function:

```
SQL> VARIABLE id NUMBER
SQL> BEGIN
  2   :id := emp_management.hire
  3   ('BLAKE', 'MANAGER', 'KING', 2990, 'SALES');
  4 END;
```

The bind variable named id can be displayed with the PRINT command or used in subsequent PL/SQL subprograms.

The following example illustrates automatically displaying a bind variable:

```
SQL> SET AUTOPRINT ON
SQL> VARIABLE a REFCURSOR
SQL> BEGIN
  2 OPEN :a FOR SELECT * FROM DEPT ORDER BY DEPTNO;
  3 END;
  4 /
```

PL/SQL procedure successfully completed.

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

In the above example, there is no need to issue a PRINT command to display the variable.

The following example creates some variables and then lists them:

```
SQL> VARIABLE id NUMBER
SQL> VARIABLE txt CHAR (20)
SQL> VARIABLE myvar REFCURSOR
SQL> VARIABLE
variable id
datatype NUMBER

variable txt
datatype CHAR(20)

variable myvar
datatype REFCURSOR
```

The following example lists a single variable:

```
SQL> VARIABLE txt
variable txt
datatype CHAR(20)
```

The following example illustrates producing a report listing individual salaries and computing the departmental and total salary cost:

```

SQL> VARIABLE RC REFCURSOR
2 BEGIN
3     OPEN :RC FOR SELECT DNAME, ENAME, SAL
4         FROM EMP, DEPT
5         WHERE EMP.DEPTNO = DEPT.DEPTNO
6         ORDER BY EMP.DEPTNO, ENAME;
7 END;
8 /

```

PL/SQL procedure successfully completed.

```

SQL> SET PAGESIZE 100 FEEDBACK OFF
SQL> TTITLE LEFT '*** Departmental Salary Bill ***' SKIP 2
SQL> COLUMN SAL FORMAT $999,990.99 HEADING 'Salary'
SQL> COLUMN DNAME HEADING 'Department'
SQL> COLUMN ENAME HEADING 'Employee'
SQL> COMPUTE SUM LABEL 'Subtotal:' OF SAL ON DNAME
SQL> COMPUTE SUM LABEL 'Total:' OF SAL ON REPORT
SQL> BREAK ON DNAME SKIP 1 ON REPORT SKIP 1
SQL> PRINT RC

```

*** Departmental Salary Bill ***

Department	Employee	Salary
ACCOUNTING	CLARK	\$2,450.00
	KING	\$5,000.00
	MILLER	\$1,300.00
*****		-----
Subtotal:		\$8,750.00
RESEARCH	ADAMS	\$1,100.00
	FORD	\$3,000.00
	JONES	\$2,975.00
	SCOTT	\$3,000.00
	SMITH	\$800.00
*****		-----
Subtotal:		\$10,875.00
SALES	ALLEN	\$1,600.00
	BLAKE	\$2,850.00
	JAMES	\$950.00
	MARTIN	\$1,250.00
	TURNER	\$1,500.00
	WARD	\$1,250.00
*****		-----
Subtotal:		\$9,400.00

Total:		\$29,025.00

The following example illustrates producing a report containing a CLOB column, and then displaying it with the SET LOBOFFSET command.

Assume you have already created a table named clob_tab which contains a column named clob_col of type CLOB. The clob_col contains the following data:

Remember to run the Departmental Salary Bill report each month. This report contains confidential information.

To produce a report listing the data in the col_clob column, enter

```
SQL> variable t clob
SQL> begin
  2   select clob_col into t: from clob_tab;
  3   end;
  4   /
PL/SQL procedure successfully completed
```

To print 200 characters from the column clob_col, enter

```
SQL> set LONG 200
SQL> print t
```

The following output results:

```
T
-----
Remember to run the Departmental Salary Bill report each month. This report
contains confidential information.
```

To set the printing position to the 21st character, enter

```
SQL> set LOBOFFSET 21
SQL> print t
```

the following output results:

```
T
-----
Departmental Salary Bill report each month. This report contains confidential
information.
```

For more information on creating CLOB columns, see your *[Oracle8i SQL Reference](#)*.

WHENEVER OSERROR

Purpose

Exits SQL*Plus if an operating system error occurs (such as a file I/O error).

Syntax

```
WHENEVER OSERROR
  {EXIT [SUCCESS|FAILURE|n|variable|:BindVariable][COMMIT|ROLLBACK]
  |CONTINUE [COMMIT|ROLLBACK|NONE]}
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

EXIT [**SUCCESS**|**FAILURE**|*n*|*variable*|:*BindVariable*]

Directs SQL*Plus to exit as soon as an operating system error is detected. You can also specify that SQL*Plus return a success or failure code, the operating system failure code, or a number or variable of your choice. See [EXIT](#) in this chapter for details.

CONTINUE

Turns off the EXIT option.

COMMIT

Directs SQL*Plus to execute a COMMIT before exiting or continuing and save pending changes to the database.

ROLLBACK

Directs SQL*Plus to execute a ROLLBACK before exiting or continuing and abandon pending changes to the database.

NONE

Directs SQL*Plus to take no action before continuing.

Usage Notes

If you do not enter the WHENEVER OSERROR command, the default behavior of SQL*Plus is to continue and take no action when an operating system error occurs.

If you do not enter the WHENEVER SQLERROR command, the default behavior of SQL*Plus is to continue and take no action when a SQL error occurs.

Examples

The commands in the following command file cause SQL*Plus to exit and COMMIT any pending changes if a failure occurs when writing to the output file:

```
SQL> WHENEVER OSERROR EXIT
SQL> START no_such_file
OS Message: No such file or directory
Disconnected from Oracle8.....
```

WHENEVER SQLERROR

Purpose

Exits SQL*Plus if a SQL command or PL/SQL block generates an error.

Syntax

```
WHENEVER SQLERROR
{EXIT [SUCCESS|FAILURE|WARNING|n|variable|:BindVariable]
[COMMIT|ROLLBACK]|CONTINUE [COMMIT|ROLLBACK|NONE]}
```

Terms and Clauses

Refer to the following list for a description of each term or clause:

EXIT [SUCCESS|FAILURE|WARNING|n|variable|:BindVariable]

Directs SQL*Plus to exit as soon as it detects a SQL command or PL/SQL block error (but after printing the error message). SQL*Plus will not exit on a SQL*Plus error. The EXIT clause of WHENEVER SQLERROR follows the same syntax as the EXIT command. See EXIT in this chapter for details.

CONTINUE

Turns off the EXIT option.

COMMIT

Directs SQL*Plus to execute a COMMIT before exiting or continuing and save pending changes to the database.

ROLLBACK

Directs SQL*Plus to execute a ROLLBACK before exiting or continuing and abandon pending changes to the database.

NONE

Directs SQL*Plus to take no action before continuing.

Usage Notes

The WHENEVER SQLERROR command is triggered by SQL command or PL/SQL block errors, and not by SQL*Plus command errors.

Examples

The commands in the following command file cause SQL*Plus to exit and return the SQL error code if the SQL UPDATE command fails:

```
SQL> WHENEVER SQLERROR EXIT SQL.SQLCODE
SQL> UPDATE EMP SET SAL = SAL*1.1
```

The following SQL command error causes SQL*Plus to exit and return the SQL error code:

```
SQL> WHENEVER SQLERROR EXIT SQL.SQLCODE
SQL> select column_does_not_exist from dual;
select column_does_not_exist from dual
      *
ERROR at line 1:
ORA-00904: invalid column name
```

Disconnected from Oracle.....

The following SQL command error causes SQL*Plus to exit and return the value of the variable `my_error_var`:

```
SQL> define my_error_var = 99
SQL> WHENEVER SQLERROR EXIT my_error_var
SQL> UPDATE non_existed_table set coll = coll + 1;
```

```
UPDATE NON_EXISTED_TABLE set coll = coll + 1
      *
```

```
ERROR at line 1:
ORA-00942: table or view does not exist
```

Disconnected from Oracle.....

The following examples show that the `WHENEVER SQLERROR` command does not have any effect on SQL*Plus commands, but does on SQL commands and PL/SQL blocks:

```
SQL> WHENEVER SQLERROR EXIT SQL.SQLCODE
SQL> column ename heading "Employee Name"
```

Unknown COLUMN option "heading"

```
SQL> show non_existed_option
```

Unknown SHOW option "non_existed_option"

```
SQL> get non_existed_file.sql
Unable to open "non_existed_file.sql"
```

The following PL/SQL block error causes SQL*Plus to exit and return the SQL error code:

```
SQL> WHENEVER SQLERROR EXIT SQL.SQLCODE
SQL> begin
  2   select column_does_not_exist from dual;
  3   end;
  4   /
```

```
select column_does_not_exist from dual;
      *
```

```
ERROR at line 2:
ORA-06550: line 2, column 10:
PLS-00201: identifier 'COLUMN_DOES_NOT_EXIST' must be declared
ORA-06550: line 2, column 3:
PL/SQL: SQL Statement ignored
```

Disconnected from Oracle.....

COLUMN DEFAULT

Purpose

Resets the display attributes for a given column to default values.

Syntax

```
COL[UMN] {column|expr} DEF[AULT]
```

Usage Notes

Has the same effect as COLUMN CLEAR.

DOCUMENT

Purpose

Begins a block of documentation in a command file.

Syntax

```
DOC[UMENT]
```

Usage Notes

For information on the current method of inserting comments in a command file, refer to the section "[Placing Comments in Command Files](#)" under "[Saving Commands for Later Use](#)" in [Chapter 3](#) and to the [REMARK](#) command in the "Command Reference" in [Chapter 8](#).

After you type DOCUMENT and enter [Return], SQL*Plus displays the prompt DOC> in place of SQL> until you end the documentation. The "pound" character (#) on a line by itself ends the documentation.

If you have set DOCUMENT to OFF, SQL*Plus suppresses the display of the block of documentation created by the DOCUMENT command. (See "[SET DOCUMENT](#)" later in this appendix.)

NEWPAGE

Purpose

Advances spooled output *n* lines beyond the beginning of the next page.

Syntax

```
NEWPAGE [1|n]
```

Usage Notes

Refer to the [NEWPAGE](#) variable of the [SET](#) command in [Chapter 8](#) for information on the current method for advancing spooled output.

SET BUFFER

Purpose

Makes the specified buffer the current buffer.

Syntax

```
SET BUF[FER] {buffer|SQL}
```

Usage Notes

Initially, the SQL buffer is the current buffer. SQL*Plus does not require the use of multiple buffers; the SQL buffer alone should meet your needs.

If the buffer name you enter does not already exist, SET BUFFER defines (creates and names) the buffer. SQL*Plus deletes the buffer and its contents when you exit SQL*Plus.

Running a query automatically makes the SQL buffer the current buffer. To copy text from one buffer to another, use the GET and SAVE commands. To clear text from the current buffer, use CLEAR BUFFER. To clear text from the SQL buffer while using a different buffer, use CLEAR SQL.

SET CLOSECURSOR

Purpose

Sets the cursor usage behavior.

Syntax

```
SET CLOSECUR[SOR] {OFF|ON}
```

Usage Notes

On or OFF sets whether or not the cursor will close and reopen after each SQL statement. This feature may be useful in some circumstances to release resources in the database server.

SET DOCUMENT

Purpose

Displays or suppresses blocks of documentation created by the DOCUMENT command.

Syntax

```
SET DOC[UMENT] {OFF|ON}
```

Usage Notes

SET DOCUMENT ON causes blocks of documentation to be echoed to the screen. Set DOCUMENT OFF suppresses the display of blocks of documentation.

See DOCUMENT in this appendix for information on the DOCUMENT command.

SET MAXDATA

Purpose

Sets the maximum total row width that SQL*Plus can process.

Syntax

```
SET MAXD[ATA] n
```

Usage Notes

In SQL*Plus, the maximum row width is unlimited. Any values you set using SET MAXDATA are ignored by SQL*Plus.

SET SCAN

Purpose

Controls scanning for the presence of substitution variables and parameters. OFF suppresses processing of substitution variables and parameters; ON allows normal processing.

Syntax

```
SET SCAN {OFF|ON}
```

Usage Notes

ON functions in the same manner as SET DEFINE ON.

SET SPACE

Purpose

Sets the number of spaces between columns in output. The maximum value of *n* is 10.

Syntax

```
SET SPACE {1|n}
```

Usage Notes

The SET SPACE 0 and SET COLSEP " commands have the same effect. This command is obsoleted by SET COLSEP, but you can still use it for backward compatibility. You may prefer to use COLSEP because the SHOW command recognizes COLSEP and does not recognize SPACE.

SET TRUNCATE

Purpose

Controls whether SQL*Plus truncates or wraps a data item that is too long for the current line width.

Syntax

```
SET TRU[NCATE] {OFF|ON}
```

Usage Notes

ON functions in the same manner as SET WRAP OFF, and vice versa. You may prefer to use WRAP because the SHOW command recognizes WRAP and does not recognize TRUNCATE.

SHOW LABEL

Purpose

Shows the security level for the current session.

Syntax

```
SHO[W] LABEL
```

Best Wishes
Sameh Bakkar