SQL*Plus

User's Guide and Reference

Release 9.0.1

June 2001

Part No. A88827-01

ORACLE

SQL*Plus User's Guide and Reference, Release 9.0.1

Part No. A88827-01

Copyright © 1996, 2001, Oracle Corporation. All rights reserved.

Primary Author: Simon Watt

Contributors: Andrew Code, Alison Goggin, Alison Holloway, Christopher Jones, Anita Lam, Luan Nim, Andrei Souleimanian, Christopher Tan, Ian Wu.

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and Oracle Net, Oracle HTTP Server, Oracle7, Oracle8, Oracle8*i*, Oracle9*i*, PL/SQL, *i*SQL*Plus and SQL*Plus are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments xi

Preface xiii

Audience xiv
Organization xiv
Related Documentation xvi
Conventions xvii
Documentation Accessibility xx

Part I Understanding SQL*Plus

1 Introduction

Overview of SQL*Plus 1-2
Basic Concepts 1-2
Who Can Use SQL*Plus 1-2
Using this Guide 1-3
Sample Tables 1-3
What You Need to Run SQL*Plus 1-4
Hardware and Software 1-4
Information Specific to Your Operating System 1-5
Username and Password 1-5
Access to Sample Tables 1-6

2 Learning SQL*Plus Basics

Starting SQL*Plus 2-2
Leaving SQL*Plus 2-3
Entering and Executing Commands 2-4
Running SQL Commands 2-6
Running PL/SQL Blocks 2-10
Running SQL*Plus Commands 2-11
Variables that Affect Running Commands 2-13
Saving Changes to the Database Automatically 2-14
Stopping a Command while it is Running 2-15
Collecting Timing Statistics on Commands You Run 2-15
Running Host Operating System Commands 2-16
Getting Help 2-16
Listing a Table Definition 2-17
Listing PL/SQL Definitions 2-18
Controlling the Display 2-18
Interpreting Error Messages 2-18

3 Manipulating Commands

Editing Commands 3-2
Listing the Buffer Contents 3-3
Editing the Current Line 3-4
Adding a New Line 3-5
Appending Text to a Line 3-6
Deleting Lines 3-7
Editing Commands with a System Editor 3-8
Saving Commands for Later Use 3-8
Storing Commands in Command Files 3-8
Placing Comments in Command Files 3-12
Retrieving Command Files 3-16
Running Command Files 3-17
Nesting Command Files 3-18
Modifying Command Files 3-19
Exiting from a Command File with a Return Code 3-20
Setting Up Your SQL*Plus Environment 3-20

Storing and Restoring SQL*Plus System Variables ... 3-21
Writing Interactive Commands ... 3-22
Defining User Variables ... 3-22
Using Substitution Variables ... 3-23
Passing Parameters through the START Command ... 3-29
Communicating with the User ... 3-30
Using Bind Variables ... 3-33
Creating Bind Variables ... 3-34
Referencing Bind Variables ... 3-34
Displaying Bind Variables ... 3-35
Using REFCURSOR Bind Variables ... 3-35
Tracing Statements ... 3-39
Controlling the Report ... 3-39
Execution Plan ... 3-40
Statistics ... 3-41
Tracing Parallel and Distributed Queries ... 3-44

4 Formatting Query Results

Formatting Columns ... 4-2
Changing Column Headings ... 4-2
Formatting NUMBER Columns ... 4-4
Formatting Datatypes ... 4-6
Copying Column Display Attributes ... 4-8
Listing and Resetting Column Display Attributes ... 4-9
Suppressing and Restoring Column Display Attributes ... 4-9
Printing a Line of Characters after Wrapped Column Values ... 4-10
Clarifying Your Report with Spacing and Summary Lines ... 4-11
Suppressing Duplicate Values in Break Columns ... 4-12
Inserting Space when a Break Column's Value Changes ... 4-13
Inserting Space after Every Row ... 4-14
Using Multiple Spacing Techniques ... 4-14
Listing and Removing Break Definitions ... 4-15
Computing Summary Lines when a Break Column's Value Changes ... 4-16
Computing Summary Lines at the End of the Report ... 4-19
Computing Multiple Summary Values and Lines ... 4-20

Listing and Removing COMPUTE Definitions 4-22
Defining Page and Report Titles and Dimensions 4-22
Setting the Top and Bottom Titles and Headers and Footers 4-22
Displaying the Page Number and other System-Maintained Values in Titles 4-27
Listing, Suppressing, and Restoring Page Title Definitions 4-28
Displaying Column Values in Titles 4-29
Displaying the Current Date in Titles 4-30
Setting Page Dimensions 4-31
Storing and Printing Query Results 4-33
Sending Results to a File 4-34
Sending Results to a Printer 4-34
Creating Web Reports 4-37
Creating Static Web Reports 4-37
Creating Dynamic Web Reports with CGI Scripts 4-42
Suppressing the Display of SQL*Plus Commands in Web Reports 4-47
HTML Entities 4-47

5 Database Administration

Overview 5-2
Introduction to Database Startup and Shutdown 5-2
Database Startup 5-2
Database Shutdown 5-3
Redo Log Files 5-4
ARCHIVELOG Mode 5-4
Database Recovery 5-5

6 Accessing SQL Databases

Connecting to the Default Database 6-2
Connecting to a Remote Database 6-3
Connecting to a Remote Database from within SQL*Plus 6-3
Connecting to a Remote Database as You Start SQL*Plus 6-4
Copying Data from One Database to Another 6-4
Understanding COPY Command Syntax 6-5
Controlling Treatment of the Destination Table 6-6
Interpreting the Messages that COPY Displays 6-8

Specifying Another User's Table 6-8
Copying Data between Tables on One Database 6-9

Part II Reference

7 Starting SQL*Plus and Getting Help

Starting SQL*Plus Using the SQLPLUS Command 7-2
Options 7-2
Logon 7-9
Start 7-10
Setting Up the Site Profile 7-10
Setting Up the User Profile 7-10
Receiving a Return Code 7-10
Getting Help 7-12

8 Command Reference

SQL*Plus Command Summary 8-2
@ ("at" sign) 8-5
@@ (double "at" sign) 8-7
/ (slash) 8-9
ACCEPT 8-10
APPEND 8-12
ARCHIVE LOG 8-13
ATTRIBUTE 8-16
BREAK 8-18
BTITLE 8-23
CHANGE 8-24
CLEAR 8-27
COLUMN 8-29
COMPUTE 8-40
CONNECT 8-46
COPY 8-48
DEFINE 8-52
DEL 8-54

DESCRIBE 8-56
DISCONNECT 8-62
EDIT 8-63
EXECUTE 8-65
EXIT 8-66
GET 8-68
HELP 8-69
HOST 8-70
INPUT 8-72
LIST 8-74
PASSWORD 8-76
PAUSE 8-77
PRINT 8-78
PROMPT 8-79
RECOVER 8-80
REMARK 8-86
REPFOOTER 8-87
REPHEADER 8-89
RUN 8-93
SAVE 8-94
SET 8-96
SHOW 8-122
SHUTDOWN 8-127
SPOOL 8-129
START 8-130
STARTUP 8-132
STORE 8-135
TIMING 8-136
TTITLE 8-138
UNDEFINE 8-142
VARIABLE 8-143
WHENEVER OSERROR 8-150
WHENEVER SQLERROR 8-152

A SQL*Plus Error Messages

B Release 9.0.1 Enhancements

C SQL*Plus Limits

D SQL Command List

E Security

F Obsolete SQL*Plus Commands

Glossary

Index

Send Us Your Comments

SQL*Plus User's Guide and Reference, Release 9.0.1

Part No. A88827-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: sqlplus@oracle.com
- FAX: +61 3 9690 0043 Attn: SQL*Plus Documentation Manager
- Postal service:
  SQL*Plus Documentation Manager
  Australian Product Development Centre
  Oracle Corporation Australia Pty Ltd
  324 St Kilda Road
  Melbourne, VIC 3004
  Australia

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

The *SQL*Plus* (pronounced “sequel plus”) *User’s Guide and Reference* introduces the SQL*Plus program and its uses. It also provides a detailed description of each SQL*Plus command.

This preface contains these topics:

- Audience
- Organization
- Related Documentation
- Conventions
- Documentation Accessibility

Audience

SQL*Plus User's Guide and Reference is intended for business and technical end users and system administrators who perform the following tasks:

- Enter, edit, store, retrieve, and run SQL commands and PL/SQL blocks
- Format, perform calculations on, store, print and create web output of query results
- List column definitions for any table
- Send messages to and accept responses from an end user
- Perform database administration

To use this document, you need a basic understanding of the SQL database language. If you do not have any familiarity with this database tool, you should refer to the *Oracle9i SQL Reference*. If you plan to use the PL/SQL database language in conjunction with SQL*Plus, refer to the *PL/SQL User's Guide and Reference* for information on using PL/SQL.

Organization

This document contains:

PART I, Understanding SQL*Plus

Contains SQL*Plus user guide and tutorial content.

Chapter 1, "Introduction"

An overview of SQL*Plus, with instructions on using this guide, and information on what you need to run SQL*Plus.

Chapter 2, "Learning SQL*Plus Basics"

Explains how to start SQL*Plus and enter and execute commands. You learn by following step-by-step examples using sample tables.

Chapter 3, "Manipulating Commands"

Contains further examples to help you learn to edit commands, save them for later use, and write interactive commands.

Chapter 4, "Formatting Query Results"

Uses examples to explain how you can format columns, clarify your reports with spacing and summary lines, define page dimensions and titles, store and print query results, and output query results to the web.

Chapter 5, "Database Administration"

Intended for use by Database Administrators (DBAs). It covers basic database administration features in SQL*Plus.

Chapter 6, "Accessing SQL Databases"

Explains how to connect to default and remote databases, and how to copy data between databases and between tables on the same database.

PART II, Reference

Contain SQL*Plus Command Reference and Appendixes.

Chapter 7, "Starting SQL*Plus and Getting Help"

Explains how to access SQL*Plus from the operating system prompt, and how to access online help.

Chapter 8, "Command Reference"

Provides a summary of SQL*Plus commands and detailed descriptions of each SQL*Plus command in alphabetical order.

Appendix A, "SQL*Plus Error Messages"

Lists messages generated by SQL*Plus, including COPY command error messages. It explains their causes, and appropriate actions for error recovery.

Appendix B, "Release 9.0.1 Enhancements"

Lists new features and enhancements for this release.

Appendix C, "SQL*Plus Limits"

Lists the maximum values for elements of SQL*Plus.

Appendix D, "SQL Command List"

Lists the major SQL commands and clauses.

Appendix E, "Security"
Explains how to restrict access to certain SQL*Plus and SQL commands.

Appendix F, "Obsolete SQL*Plus Commands"
Provides information on Obsolete SQL*Plus commands.

Glossary
Defines technical terms associated with Oracle and SQL*Plus.

Related Documentation

For more information, see these Oracle resources:

- *SQL*Plus Quick Reference*
- *iSQL*Plus User's Guide and Reference*
- *PL/SQL User's Guide and Reference*
- *Oracle9i SQL Reference*
- *Oracle9i Concepts*
- *Oracle9i Administrator's Guide*
- *Oracle9i User-Managed Backup and Recovery Guide*
- *Oracle9i Application Developer's Guide - Fundamentals*
- *Oracle9i Heterogeneous Services*
- *Oracle9i Replication*
- *Oracle9i Utilities*
- *Oracle9i Error Messages*
- *Oracle9i Migration*
- *Oracle9i Reference*
- *Oracle9i Performance Guide and Reference*
- *Oracle9i Real Application Clusters Concepts*
- *Oracle Net Services Administrator's Guide*
- *Oracle Call Interface Programmer's Guide*
- *Pro*COBOL Precompiler Programmer's Guide*

- *Pro*C/C++ Precompiler Programmer's Guide*
- Oracle installation and user's manual(s) provided for your operating system

Many of the examples in this book use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle9i Sample Schemas* for information on how these schemas were created and how you can use them yourself.

In North America, printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://technet.oracle.com/membership/index.htm`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://technet.oracle.com/docs/index.htm`

Whitepapers, sample code, frequently asked questions and other useful information is regularly posted to the SQL*Plus section on OTN at

`http://technet.oracle.com/tech/sql_plus/`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width font)`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width font)`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase monospace (fixed-width font) italic`	Lowercase monospace italic font represents placeholders or variables.	You can specify the *`managed_clause`*. Run *`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. If users are expected to type them into the system, they are identified by the keyboard icon shown in the margin following. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

Similarly, output from an example is identified by a computer screen icon in the margin as shown in the margin following.

```
PAGESIZE 24
```

Where both icons occur together, it implies interative entry and output.

```
1
  1* SELECT LAST_NAME, SALARY
APPEND , COMMISSION_PCT;
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
. . .	Horizontal ellipsis points indicate either:	
	■ That we have omitted parts of the code that are not directly related to the example	`CREATE TABLE ... AS `*`subquery`*`;`
	■ That you can repeat a portion of the code	`SELECT `*`col1`*`, `*`col2`*`, ... , `*`coln`*` FROM employees;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	

Convention	Meaning	Example
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`* `DB_NAME =` *`database_name`*

UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Part I

Understanding SQL*Plus

This section provides an introduction to SQL*Plus. It provides an overview of how to run SQL*Plus and demonstrates this with various examples.

The following chapters are covered in this section:

- Introduction
- Learning SQL*Plus Basics
- Manipulating Commands
- Formatting Query Results
- Database Administration
- Accessing SQL Databases

1

Introduction

This chapter introduces you to SQL*Plus, covering the following topics:

- Overview of SQL*Plus
- Using this Guide
- What You Need to Run SQL*Plus

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 execute 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, print and create web output of query results
- 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 PL/SQL commands related to one another through procedural logic.
table	The basic unit of storage in Oracle.
query	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 LAST_NAME in the database the clearer heading “Last Name”, you might enter the following command:

```
COLUMN LAST_NAME HEADING 'Last Name'
```

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

```
DESCRIBE EMPLOYEES
```

Using this Guide

This guide gives you information about SQL*Plus that applies to all operating systems. Some aspects of SQL*Plus, however, differ on each operating system. Such operating system specific details are covered in the Oracle installation and user’s guide provided for your system. Use these operating system specific guides in conjunction with the *SQL*Plus User’s Guide and Reference*.

Throughout this guide, examples showing how to enter commands use a common command syntax and a common set of sample tables. The tables are described below.

You will find the "Conventions in Code Examples" section in the Preface particularly useful when referring to commands in Chapter 7 and Chapter 8 of this guide.

Sample Tables

Included with Oracle9*i*, are a number of sample schemas. The tutorial and examples in this guide use the EMP_DETAILS_VIEW view of the Human Resources (HR) sample schema. In using the HR sample schema you will come to understand the concepts and operations of this guide. This schema contains personnel records for a fictitious company. As you complete the exercises in this guide, imagine that you are the personnel director for this company.

Note: Dates in the sample tables use four digit years. As the default date format in SQL*Plus is DD-MM-YY, dates displayed show only a two digit year. Use the SQL TO_CHAR function in your SELECT statements to control the way dates are displayed.

For further information about the sample schemas included with Oracle9*i*, see the *Oracle9i Sample Schemas* guide. Figure 1–1 shows a description of the view, EMP_DETAILS_VIEW.

Figure 1–1 EMP_DETAILS_VIEW

```
Name                                    Null?    Type
----------------------------------------- -------- ----------------------------
EMPLOYEE_ID                             NOT NULL NUMBER(6)
JOB_ID                                  NOT NULL VARCHAR2(10)
MANAGER_ID                                       NUMBER(6)
DEPARTMENT_ID                                    NUMBER(4)
LOCATION_ID                                      NUMBER(4)
COUNTRY_ID                                       CHAR(2)
FIRST_NAME                                       VARCHAR2(20)
LAST_NAME                               NOT NULL VARCHAR2(25)
SALARY                                           NUMBER(8,2)
COMMISSION_PCT                                   NUMBER(2,2)
DEPARTMENT_NAME                         NOT NULL VARCHAR2(30)
JOB_TITLE                               NOT NULL VARCHAR2(35)
CITY                                    NOT NULL VARCHAR2(30)
STATE_PROVINCE                                   VARCHAR2(25)
COUNTRY_NAME                                     VARCHAR2(40)
REGION_NAME                                      VARCHAR2(25)
```

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.

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 a DBA, see the instructions for installing Oracle and SQL*Plus in the Oracle installation and user's guide 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 guide, 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 guide available for reference.

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. Default logins are created and displayed in messages during Oracle9*i* installation. The default login *username*/*password* combinations created are:

- SYS/CHANGE_ON_INSTALL
- SYSTEM/MANAGER
- HR/<YOUR_SECRET_PASSWORD>

Default passwords should be changed as soon as possible. See the PASSWORD command in Chapter 8 for details on how to change your password.

For further information about the default logins, see the *Oracle9i Administrator's Guide*.

Multi-User Systems

Each user must have a username and password to access 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 functions of a DBA for yourself. If you want to define your own username and password, see the *Oracle9i SQL Reference*.

Access to Sample Tables

The Human Resources (HR) Sample Schema is installed as part of the default Oracle9*i* installation. The HR user is locked by default.

To use the HR sample schema, you need to unlock the HR tables and user. To unlock the HR tables and user, log in to SQL*Plus as the SYSTEM user and enter the following command:

```
ALTER USER HR IDENTIFIED BY HR ACCOUNT UNLOCK;
```

For further information about unlocking the HR / HR tables and login, see the *Oracle9i Sample Schemas* guide. The HR / HR user is primarily to enable you to access the HR sample schema and is necessary to enable you to run the examples in this guide.

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 HR tables, see your DBA or see the *Oracle9i Sample Schemas* guide, or you can create the HR tables with the script HR_MAIN.SQL which is located in the following subdirectory:

```
$ORACLE_HOME/DEMO/SCHEMA/HUMAN_RESOURCES/HR_MAIN.SQL
```

When you have no further use for the sample tables, remove them by running another Oracle supplied command file named HR_DROP.SQL. For instructions on how to run these files, see the Oracle installation and users guide provided for your operating system, and the *Oracle9i Sample Schemas* guide.

2

Learning SQL*Plus Basics

This chapter helps you learn the basics of using SQL*Plus, including the following topics:

- Starting SQL*Plus
- Leaving SQL*Plus
- Entering and Executing Commands
- Getting Help

Read this chapter while sitting at your computer and try out the examples shown. Before beginning, make sure you have access to the sample tables described in Chapter 1, "Introduction".

Starting SQL*Plus

To begin using SQL*Plus, you must first understand how to start and leave SQL*Plus.

Example 2–1 Starting SQL*Plus

This example shows you how to start SQL*Plus. Follow the steps shown.

1. Make sure that SQL*Plus has been installed on your computer.
2. Turn on your computer (if it is off) and log on to the host operating system (if required). If you are already using your computer, you need not log off or reset it. Simply exit from the program you are using (if any).

   You should see one or more characters at the left side of the screen. This is the operating system's command prompt, which signals that the operating system is ready to accept a command. In this guide the operating system's prompt will be represented by a dollar sign ($). Your computer's operating system prompt may be different.
3. Enter the command SQLPLUS and press Return. This is an operating system command that starts SQL*Plus.

   Note: Some operating systems expect you to enter commands in lowercase letters. If your system expects lowercase, enter the SQLPLUS command in lowercase.

```
SQLPLUS
```

SQL*Plus displays its version number, the current date, and copyright information, and prompts you for your username (the text displayed on your system may differ slightly):

```
SQL*Plus: Release 9.0.1.0.0 - Production on Thu June 14 16:29:01 2001
(c) Copyright 1996, 2001 Oracle Corporation. All rights reserved.
Enter user-name:
```

4. Enter your username and press Return. SQL*Plus displays the prompt "Enter password:".
5. Enter your password and press Return again. For your protection, your password does not appear on the screen.

The process of entering your username and password is called *logging in*. SQL*Plus displays the version of Oracle to which you connected and the versions of available tools such as PL/SQL.

Next, SQL*Plus displays the SQL*Plus command prompt:

```
SQL>
```

The command prompt indicates that SQL*Plus is ready to accept your commands. Throughout this guide, where you see the following keyboard icon in the margin, it is prompting you to enter information at the command prompt line. Where you see the computer screen icon in the margin, it is showing you what you should expect to see displayed on your screen.

If SQL*Plus does not start, you should see a message to help you correct the problem.

Shortcuts to Starting SQL*Plus

When you start SQL*Plus, you can enter your username and password, separated by a slash (/), following the command SQLPLUS. For example, if your username is HR and your password is HR, you can enter

```
SQLPLUS HR/HR
```

and press Return. You can also arrange to log in to SQL*Plus automatically when you log on to your host operating system. See the Oracle installation and user's guide provided for your operating system for details.

Leaving SQL*Plus

When you are done working with SQL*Plus and wish to return to the operating system, enter the EXIT command at the SQL*Plus command prompt.

*Example 2–2 Exiting SQL*Plus*

To leave SQL*Plus, enter the EXIT command at the SQL*Plus command prompt:

```
EXIT
```

SQL*Plus displays the version of Oracle from which you disconnected and the versions of tools available through SQL*Plus. After a moment you will see the operating system prompt.

Before continuing with this chapter, follow steps 3, 4, and 5 of Example 2–1 to start SQL*Plus again. Alternatively, log in using the shortcut shown under "Shortcuts to Starting SQL*Plus" above.

Entering and Executing Commands

Entering Commands

Your computer's cursor, or pointer (typically an underline, a rectangular block, or a slash), appears after the command prompt. The cursor indicates the place where the next character you type will appear on your screen.

To tell SQL*Plus what to do, simply type the command you wish to use. Usually, you separate the words in a command from each other by a space or tab. You can use additional spaces or tabs between words, if you wish, to make your commands more readable.

Note: You will see examples of spacing and indentation throughout this guide. When you enter the commands in the exercises, you do not have to space them as shown, but you may find them clearer to read if you do.

Case sensitivity is operating system specific. For the sake of clarity, all table names, column names, and commands in this guide appear in capital letters.

You can enter three kinds of commands at the command prompt:

- SQL commands, for working with information in the database
- PL/SQL blocks, also for working with information in the database
- SQL*Plus commands, for formatting query results, setting options, and editing and storing SQL commands and PL/SQL blocks

The manner in which you continue a command on additional lines, end a command, or execute a command differs depending on the type of command you wish to enter and run. Examples of how to run and execute these types of commands are found on the following pages.

The SQL Buffer

The area where SQL*Plus stores your most recently entered SQL command or PL/SQL block is called the *SQL buffer*. The command or block remains there until you enter another. If you want to edit or re-run the current SQL command or PL/SQL block, you may do so without re-entering it. For more information about editing or re-running a command or block stored in the buffer see the section "Running Command Files" in Chapter 3.

SQL*Plus does not store the semicolon or the slash you type to execute a command in the SQL buffer.

Note: SQL*Plus commands are not stored in the SQL buffer.

Getting Help

The database administrator creates the SQL*Plus help tables and populates them with SQL*Plus help data. Before you can install SQL*Plus help, ensure that:

- SQL*Plus is installed, otherwise, you cannot create and load the help tables.
- The default tablespace for the SYSTEM user is large enough to accommodate the help system. You must have at least 128K of free space.
- The SQL*Plus help script files are available in

  ```
  $ORACLE_HOME/SQLPLUS/ADMIN/HELP/
  ```

The help script files are:

- HLPBLD.SQL - to drop and create new help tables.
- HELPUS.SQL - to populate the help tables with the help data.
- HELPDROP.SQL - to drop existing SQL*Plus help tables.

To install SQL*Plus help:

1. Run SQL*Plus as the SYSTEM user with:

   ```
   SQLPLUS SYSTEM/PASSWORD
   ```

   where PASSWORD is the password you have defined for the SYSTEM user (MANAGER by default).

2. Run the SQL script, HLPBLD.SQL, from SQL*Plus with:

   ```
   @$ORACLE_HOME/SQLPLUS/ADMIN/HELP/HLPBLD.SQL HELPUS.SQL
   ```

To get online help for SQL*Plus commands, type HELP at the command prompt followed by the name of the command, for example:

```
HELP ACCEPT
```

If you get a response indicating that help is not available, consult your database administrator. For more details about the help system, see "Getting Help" in Chapter 7, and the HELP command in Chapter 8.

Executing Commands

After you enter the command and direct SQL*Plus to execute it, SQL*Plus processes the command and re-displays the command prompt, indicating that you can enter another command.

Running SQL Commands

The SQL command language enables you to manipulate data in the database. See your *Oracle9i SQL Reference* for information on individual SQL commands.

Example 2–3 Entering a SQL Command

In this example, you will enter and execute a SQL command to display the employee number, name, job, and salary of each employee in the EMP_DETAILS_VIEW view.

1. At the command prompt, enter the first line of the command:

```
SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, SALARY
```

   If you make a mistake, use Backspace to erase it and re-enter. When you are done, press Return to move to the next line.

2. SQL*Plus will display a "2", the prompt for the second line. Enter the second line of the command:

```
FROM EMP_DETAILS_VIEW WHERE SALARY > 12000;
```

   The semicolon (;) means that this is the end of the command. Press Return. SQL*Plus processes the command and displays the results on the screen:

```
EMPLOYEE_ID LAST_NAME                 JOB_ID              SALARY
----------- ------------------------- ---------- --------------
        100 King                      AD_PRES           $24,000
        101 Kochhar                   AD_VP             $17,000
```

```
102 De Haan                   AD_VP           $17,000
145 Russell                   SA_MAN          $14,000
146 Partners                  SA_MAN          $13,500
201 Hartstein                 MK_MAN          $13,000

6 rows selected.
```

After displaying the results and the number of rows retrieved, SQL*Plus displays the command prompt again. If you made a mistake and therefore did not get the results shown above, simply re-enter the command.

The headings may be repeated in your output, depending on the setting of a system variable called PAGESIZE. Sometimes, the result from a query will not fit the available page width. You will need to adjust a system variable called LINESIZE, which sets the width of the output in characters, see "Setting Page Dimensions". Typically, in the examples in this guide this is set to 70 characters. You may need to SET LINESIZE to 70 so the query output appears the same as in this guide. Whether you see the message concerning the number of records retrieved depends on the setting of a system variable called FEEDBACK. You will learn more about system variables later in this chapter in the section "Variables that Affect Running Commands". To save space, the number of records selected will not be shown in the rest of the examples in this guide.

Understanding SQL Command Syntax

Just as spoken language has syntax rules that govern the way we assemble words into sentences, SQL*Plus has syntax rules that govern how you assemble words into commands. You must follow these rules if you want SQL*Plus to accept and execute your commands.

Dividing a SQL Command into Separate Lines You can divide your SQL command into separate lines at any points you wish, as long as individual words are not split between lines. Thus, you can enter the query you entered in Example 2-3 on three lines:

```
SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000;
```

In this guide, you will find most SQL commands divided into clauses, one clause on each line. In Example 2–3, for instance, the SELECT and FROM clauses were placed on separate lines. Many people find this clearly visible structure helpful, but you may choose whatever line division makes commands most readable to you.

Ending a SQL Command You can end a SQL command in one of three ways:

- with a semicolon (;)
- with a slash (/) on a line by itself
- with a blank line

A semicolon (;) tells SQL*Plus that you want to run the command. Type the semicolon at the end of the last line of the command, as shown in Example 2–3, and press Return. SQL*Plus will process the command and store it in the SQL buffer (see the section "The SQL Buffer" for details). If you mistakenly press Return before typing the semicolon, SQL*Plus prompts you with a line number for the next line of your command. Type the semicolon and press Return again to run the command.

Note: You cannot enter a comment on the same line after a semicolon. For more information about placing comments, see "Placing Comments in Command Files" in Chapter 3.

A slash (/) on a line by itself also tells SQL*Plus that you wish to run the command. Press Return at the end of the last line of the command. SQL*Plus prompts you with another line number. Type a slash and press Return again. SQL*Plus executes the command and stores it in the buffer (see the section "The SQL Buffer" for details).

A blank line in a SQL statement or script tells SQL*Plus that you have finished entering the command, but do not want to run it yet. Press Return at the end of the last line of the command. SQL*Plus prompts you with another line number.

Note: You can change the way blank lines appear and behave in SQL statements using the SET SQLBLANKLINES command. For more information about changing blank line behavior, see the SET command in Chapter 8.

Press Return again; SQL*Plus now prompts you with the SQL*Plus command prompt. SQL*Plus does not execute the command, but stores it in the SQL buffer (see the section "The SQL Buffer" for details). If you subsequently enter another SQL command, SQL*Plus overwrites the previous command in the buffer.

Creating Stored Procedures Stored procedures are PL/SQL functions, packages, or procedures. To create stored procedures, you use SQL CREATE commands. The following SQL CREATE commands are used to create stored procedures:

- CREATE FUNCTION
- CREATE LIBRARY
- CREATE PACKAGE
- CREATE PACKAGE BODY
- CREATE PROCEDURE
- CREATE TRIGGER
- CREATE TYPE

Entering any of these commands places you in PL/SQL mode, where you can enter your PL/SQL subprogram. For more information, see the section "Running PL/SQL Blocks" in this chapter). When you are done typing your PL/SQL subprogram, enter a period (.) on a line by itself to terminate PL/SQL mode. To run the SQL command and create the stored procedure, you must enter RUN or slash (/). A semicolon (;) will not execute these CREATE commands.

When you use CREATE to create a stored procedure, a message appears if there are compilation errors. To view these errors, you use SHOW ERRORS. For example:

```
SHOW ERRORS PROCEDURE ASSIGNVL
```

For more information about the SHOW command, see Chapter 8, "Command Reference".

To execute a PL/SQL statement that references a stored procedure, you can use the EXECUTE command. EXECUTE runs the PL/SQL statement that you enter immediately after the command. For example:

```
EXECUTE :ID := EMPLOYEE_MANAGEMENT.GET_ID('BLAKE')
```

For more information about the EXECUTE command, see Chapter 8, "Command Reference".

Executing the Current SQL Command or PL/SQL Block from the Command Prompt

You can run (or re-run) the current SQL command or PL/SQL block by entering the RUN command or the slash (/) command at the command prompt. The RUN command lists the SQL command or PL/SQL block in the buffer before executing the command or block; the slash (/) command simply runs the SQL command or PL/SQL block.

Running PL/SQL Blocks

You can also use PL/SQL subprograms (called *blocks*) to manipulate data in the database. See your *PL/SQL User's Guide and Reference* for information on individual PL/SQL statements.

To enter a PL/SQL subprogram in SQL*Plus, you need to be in PL/SQL mode. You are placed in PL/SQL mode when

- You type DECLARE or BEGIN at the SQL*Plus command prompt. After you enter PL/SQL mode in this way, type the remainder of your PL/SQL subprogram.
- You type a SQL command (such as CREATE FUNCTION) that creates a stored procedure. After you enter PL/SQL mode in this way, type the stored procedure you want to create.

SQL*Plus treats PL/SQL subprograms in the same manner as SQL commands, except that a semicolon (;) or a blank line does not terminate and execute a block. Terminate PL/SQL subprograms by entering a period (.) by itself on a new line. You can also terminate and execute a PL/SQL subprogram by entering a slash (/) by itself on a new line.

SQL*Plus stores the subprograms you enter at the SQL*Plus command prompt in the SQL buffer. Execute the current subprogram by issuing a RUN or slash (/) command. Likewise, to execute a SQL CREATE command that creates a stored procedure, you must also enter RUN or slash (/). A semicolon (;) will not execute these SQL commands as it does other SQL commands.

SQL*Plus sends the complete PL/SQL subprogram to Oracle for processing (as it does SQL commands). See your *PL/SQL User's Guide and Reference* for more information.

You might enter and execute a PL/SQL subprogram as follows:

```
DECLARE
    x   NUMBER := 100;
    BEGIN
        FOR i IN 1..10 LOOP
           IF MOD (i, 2) = 0 THEN    --i is even
             INSERT INTO temp VALUES (i, x, 'i is even');
           ELSE
             INSERT INTO temp VALUES (i, x, 'i is odd');
           END IF;
           x := x + 100;
       END LOOP;
   END;
   .
/
```

When you run a subprogram, the SQL commands within the subprogram may behave somewhat differently than they would outside the subprogram. See your *PL/SQL User's Guide and Reference* for detailed information on the PL/SQL language.

Running SQL*Plus Commands

You can use SQL*Plus commands to manipulate SQL commands and PL/SQL blocks and to format and print query results. SQL*Plus treats SQL*Plus commands differently than SQL commands or PL/SQL blocks. For information on individual SQL*Plus commands, see Chapter 8, "Command Reference".

To speed up command entry, you can abbreviate many SQL*Plus commands to one or a few letters. Abbreviations for some SQL*Plus commands are described along with the commands in Chapter 3, Chapter 4, and Chapter 6. For abbreviations of all SQL*Plus commands, see Chapter 8, "Command Reference".

*Example 2–4 Entering a SQL*Plus Command*

This example shows how you might enter a SQL*Plus command to change the format used to display the column SALARY of the sample view, EMP_DETAILS_VIEW.

1. On the command line, enter this SQL*Plus command:

```
COLUMN SALARY FORMAT $99,999 HEADING 'MONTHLY SALARY'
```

If you make a mistake, use Backspace to erase it and re-enter. When you have entered the line, press Return. SQL*Plus notes the new format and displays the SQL*Plus command prompt again, ready for a new command.

2. Enter the RUN command to re-run the most recent query (from Example 2-3):

```
RUN
```

```
EMPLOYEE_ID LAST_NAME                 JOB_ID     MONTHLY SALARY
----------- ------------------------- ---------- --------------
        100 King                      AD_PRES           $24,000
        101 Kochhar                   AD_VP             $17,000
        102 De Haan                   AD_VP             $17,000
        145 Russell                   SA_MAN            $14,000
        146 Partners                  SA_MAN            $13,500
        201 Hartstein                 MK_MAN            $13,000

6 rows selected.
```

The COLUMN command formatted the column SALARY with a dollar sign ($) and a comma (,) and gave it a new heading. The RUN command then re-ran the query of Example 2-3, which was stored in the buffer. SQL*Plus does not store SQL*Plus commands in the SQL buffer.

Understanding SQL*Plus Command Syntax

SQL*Plus commands have a different syntax from SQL commands or PL/SQL blocks.

Continuing a Long SQL*Plus Command on Additional Lines You can continue a long SQL*Plus command by typing a hyphen at the end of the line and pressing Return. If you wish, you can type a space before typing the hyphen. SQL*Plus displays a right angle-bracket (>) as a prompt for each additional line.

For example:

```
COLUMN SALARY FORMAT $99,999 -
HEADING 'MONTHLY SALARY'
```

Since SQL*Plus identifies the hyphen as a continuation character, entering a hyphen within a SQL statement is ignored by SQL*Plus. SQL*Plus does not identify the statement as a SQL statement until after the input processing has joined the lines together and removed the hyphen. For example, entering the following:

```
SELECT 200 -
100 FROM DUAL;
```

returns the error:

```
SELECT 200 100 FROM DUAL
           *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

To ensure that the statement is interpreted correctly, reposition the hyphen from the end of the first line to the beginning of the second line.

Ending a SQL*Plus Command You do not need to end a SQL*Plus command with a semicolon. When you finish entering the command, you can just press Return. If you wish, however, you can enter a semicolon at the end of a SQL*Plus command.

Variables that Affect Running Commands

The SQL*Plus command SET controls many variables—called *system variables*—the settings of which affect the way SQL*Plus runs your commands. System variables control a variety of conditions within SQL*Plus, including default column widths for your output, whether SQL*Plus displays the number of records selected by a command, and your page size. System variables are also called *SET command variables*.

The examples in this guide are based on running SQL*Plus with the system variables at their default settings. Depending on the settings of your system variables, your output may appear slightly different than the output shown in the examples. (Your settings might differ from the default settings if you have a SQL*Plus LOGIN file on your computer.)

For more information on system variables and their default settings, see the SET command in Chapter 8. For details on the SQL*Plus LOGIN file, refer to the section "Setting Up Your SQL*Plus Environment" under "Saving Commands for Later Use" in Chapter 3 and to the SQLPLUS command in Chapter 7.

To list the current setting of a SET command variable, enter SHOW followed by the variable name at the command prompt. See the SHOW command in Chapter 8 for information on other items you can list with SHOW.

Saving Changes to the Database Automatically

Through the SQL DML commands UPDATE, INSERT, and DELETE—which can be used independently or within a PL/SQL block—specify changes you wish to make to the information stored in the database. These changes are not made permanent until you enter a SQL COMMIT command or a SQL DCL or DDL command (such as CREATE TABLE), or use the autocommit feature. The SQL*Plus autocommit feature causes pending changes to be committed after a specified number of successful SQL DML transactions. (A SQL DML transaction is either an UPDATE, INSERT, or DELETE command, or a PL/SQL block.)

You control the autocommit feature with the SQL*Plus SET command's AUTOCOMMIT variable.

Example 2–5 Turning Autocommit On

To turn the autocommit feature on, enter

```
SET AUTOCOMMIT ON
```

Alternatively, you can enter the following to turn the autocommit feature on:

```
SET AUTOCOMMIT IMMEDIATE
```

Until you change the setting of AUTOCOMMIT, SQL*Plus automatically commits changes from each SQL DML command that specifies changes to the database. After each autocommit, SQL*Plus displays the following message:

```
COMMIT COMPLETE
```

When the autocommit feature is turned on, you cannot roll back changes to the database.

To commit changes to the database after a number of SQL DML commands, for example, 10, enter

```
SET AUTOCOMMIT 10
```

SQL*Plus counts SQL DML commands as they are executed and commits the changes after each 10th SQL DML command.

> **Note:** For this feature, a PL/SQL block is regarded as one transaction, regardless of the actual number of SQL commands contained within it.

To turn the autocommit feature off again, enter the following command:

```
SET AUTOCOMMIT OFF
```

To confirm that AUTOCOMMIT is now set to OFF, enter the following SHOW command:

```
SHOW AUTOCOMMIT
```

```
AUTOCOMMIT OFF
```

For more information, see the AUTOCOMMIT variable of the SET command in Chapter 8.

Stopping a Command while it is Running

Suppose you have displayed the first page of a 50 page report and decide you do not need to see the rest of it. Press Cancel, the system's interrupt character, which is usually CTRL+C. SQL*Plus stops the display and returns to the command prompt.

Note: Pressing Cancel does not stop the printing of a file that you have sent to a printer with the OUT clause of the SQL*Plus SPOOL command. (You will learn about printing query results in Chapter 4.) You can stop the printing of a file through your operating system. For more information, see your operating system's installation and user's guide.

Collecting Timing Statistics on Commands You Run

Use the SQL*Plus TIMING command to collect and display data on the amount of computer resources used to run one or more commands or blocks. TIMING collects data for an elapsed period of time, saving the data on commands run during the period in a timer.

See TIMING in Chapter 8 and the Oracle installation and user's guide provided for your operating system for more information. See also "Tracing Statements" in Chapter 3 for information about using AUTOTRACE to collect statistics.

To delete all timers, enter CLEAR TIMING at the command prompt.

Running Host Operating System Commands

You can execute a host operating system command from the SQL*Plus command prompt. This is useful when you want to perform a task such as listing existing host operating system files.

To run a host operating system command, enter the SQL*Plus command HOST followed by the host operating system command. For example, this SQL*Plus command runs a host command, DIRECTORY *.SQL:

```
HOST DIRECTORY *.SQL
```

When the host command finishes running, the SQL*Plus command prompt appears again.

Note: Operating system commands entered from a SQL*Plus session using the HOST command do not effect the current SQL*Plus session. For example, setting an operating system environment variable does not effect the current SQL*Plus session, but may effect SQL*Plus sessions started subsequently.

You can suppress access to the HOST command. For more information about suppressing the HOST command see Appendix E, "Security".

Getting Help

While you use SQL*Plus, you may find that you need to list column definitions for a table, or start and stop the display that scrolls by. You may also need to interpret error messages you receive when you enter a command incorrectly or when there is a problem with Oracle or SQL*Plus. The following sections describe how to get help for those situations.

Listing a Table Definition

To see the definitions of each column in a given table or view, use the SQL*Plus DESCRIBE command.

Example 2–6 Using the DESCRIBE Command

To list the column definitions of the columns in the sample view EMP_DETAILS_VIEW, enter

```
DESCRIBE EMP_DETAILS_VIEW;
```

```
 Name                                    Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE_ID                             NOT NULL NUMBER(6)
 JOB_ID                                  NOT NULL VARCHAR2(10)
 MANAGER_ID                                       NUMBER(6)
 DEPARTMENT_ID                                    NUMBER(4)
 LOCATION_ID                                      NUMBER(4)
 COUNTRY_ID                                       CHAR(2)
 FIRST_NAME                                       VARCHAR2(20)
 LAST_NAME                               NOT NULL VARCHAR2(25)
 SALARY                                           NUMBER(8,2)
 COMMISSION_PCT                                   NUMBER(2,2)
 DEPARTMENT_NAME                         NOT NULL VARCHAR2(30)
 JOB_TITLE                               NOT NULL VARCHAR2(35)
 CITY                                    NOT NULL VARCHAR2(30)
 STATE_PROVINCE                                   VARCHAR2(25)
 COUNTRY_NAME                                     VARCHAR2(40)
 REGION_NAME                                      VARCHAR2(25)
```

Note: DESCRIBE accesses information in the Oracle data dictionary. You can also use SQL SELECT commands to access this and other information in the database. See your *Oracle9i SQL Reference* for details.

Listing PL/SQL Definitions

To see the definition of a function or procedure, use the SQL*Plus DESCRIBE command.

Example 2–7 Using the DESCRIBE Command

To list the definition of a function called AFUNC, enter

```
DESCRIBE afunc
```

```
FUNCTION afunc RETURNS NUMBER
Argument Name   Type     In/Out   Default?
--------------- -------- -------- ---------
F1              CHAR     IN
F2              NUMBER   IN
```

Controlling the Display

Suppose that you wish to stop and examine the contents of the screen while displaying a long report or the definition of a table with many columns. The display will pause while you examine it. To continue, press Resume.

If you wish, you can use the PAUSE variable of the SQL*Plus SET command to have SQL*Plus pause after displaying each screen of a query or report. For more information, refer to the SET command in Chapter 8.

Interpreting Error Messages

If SQL*Plus detects an error in a command, it displays an error message. See Appendix A, "SQL*Plus Error Messages" for a list of SQL*Plus error messages.

Example 2–8 Interpreting an Error Message

If you attempt to execute a file that does not exist or is unavailable by entering:

```
START EMPLYYES.SQL
```

An error message indicates that the table does not exist:

```
SP2-0310: unable to open file "emplyyes.sql"
```

You will often be able to figure out how to correct the problem from the message alone. If you need further explanation, take one of the following steps to determine the cause of the problem and how to correct it:

- If the error is a numbered error beginning with the letters “SP2”, look up the SQL*Plus message in Appendix A, "SQL*Plus Error Messages" of this guide.
- If the error is a numbered error beginning with the letters “CPY" look up the SQL*Plus COPY command message in Appendix A, "SQL*Plus Error Messages" of this guide.
- If the error is a numbered error beginning with the letters “ORA”, look up the Oracle message in the *Oracle9i Error Messages* guide or in the Oracle installation and user’s guides provided for your operating system.

If the error is unnumbered, look up correct syntax for the command that generated the error in Chapter 8, "Command Reference" of this guide for a SQL*Plus command, in the *Oracle9i SQL Reference* for a SQL command, or in the *PL/SQL User’s Guide and Reference* for a PL/SQL block. Otherwise, contact your DBA.

3

Manipulating Commands

This chapter helps you learn to manipulate SQL*Plus commands, SQL commands, and PL/SQL blocks. It covers the following topics:

- Editing Commands
- Saving Commands for Later Use
- Writing Interactive Commands
- Using Bind Variables
- Tracing Statements

Read this chapter while sitting at your computer and try out the examples shown. Before beginning, make sure you have access to the sample schema described in Chapter 1.

Editing Commands

Because SQL*Plus does not store SQL*Plus commands in the buffer, you edit a SQL*Plus command entered directly to the command prompt by using Backspace or by re-entering the command.

You can use a number of SQL*Plus commands to edit the SQL command or PL/SQL block currently stored in the buffer. Alternatively, you can use a host operating system editor to edit the buffer contents.

Table 3–1 lists the SQL*Plus commands that allow you to examine or change the command in the buffer without re-entering the command.

Table 3–1 SQL*Plus Editing Commands

Command	Abbreviation	Purpose
APPEND *text*	A *text*	adds *text* at the end of a line
CHANGE /old/new	C */old/new*	changes *old* to *new* in a line
CHANGE */text*	C */text*	deletes *text* from a line
CLEAR BUFFER	CL BUFF	deletes all lines
DEL	(none)	deletes the current line
DEL *n*	(none)	deletes line *n*
DEL *	(none)	deletes the current line
DEL *n* *	(none)	deletes line *n* through the current line
DEL LAST	(none)	deletes the last line
DEL *m n*	(none)	deletes a range of lines (*m* to *n*)
DEL * *n*	(none)	deletes the current line through line *n*
INPUT	I	adds one or more lines
INPUT *text*	I *text*	adds a line consisting of *text*
LIST	L	lists all lines in the SQL buffer
LIST *n*	L *n* or *n*	lists line *n*
LIST *	L *	lists the current line

*Table 3–1 SQL*Plus Editing Commands*

Command	Abbreviation	Purpose
LIST *n* *	L *n* *	lists line *n* through the current line
LIST LAST	L LAST	lists the last line
LIST *m n*	L *m n*	lists a range of lines (*m* to *n*)
LIST * *n*	L * *n*	lists the current line through line *n*

You will find these commands useful if you mis-type a command or wish to modify a command you have entered.

Listing the Buffer Contents

Any editing command other than LIST and DEL affects only a single line in the buffer. This line is called the *current line*. It is marked with an asterisk when you list the current command or block.

Example 3–1 Listing the Buffer Contents

Suppose you want to list the current command. Use the LIST command as shown below. (If you have EXITed SQL*Plus or entered another SQL command or PL/SQL block since following the steps in Example 2–3, perform the steps in that example again before continuing.)

```
LIST
```

```
SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, SALARY
  2  FROM EMP_DETAILS_VIEW
  3* WHERE SALARY>12000
```

Notice that the semicolon you entered at the end of the SELECT command is not listed. This semicolon is necessary to mark the end of the command when you enter it, but SQL*Plus does not store it in the SQL buffer. This makes editing more convenient, since it means you can append a new line to the end of the buffer without removing a semicolon.

Editing the Current Line

The SQL*Plus CHANGE command allows you to edit the current line. Various actions determine which line is the current line:

- LIST a given line to make it the current line.
- When you LIST or RUN the command in the buffer, the last line of the command becomes the current line. (Note, that using the slash (/) command to run the command in the buffer does not affect the current line.)
- If you get an error message, the line containing the error automatically becomes the current line.

Example 3–2 Making an Error in Command Entry

Suppose you try to select the JOB_ID column but mistakenly enter it as JO_ID. Enter the following command, purposely misspelling JOB_ID in the first line:

```
SELECT EMPLOYEE_ID, LAST_NAME, JO_ID, SALARY
FROM EMP_DETAILS_VIEW
WHERE JOB_ID='SA_MAN';
```

You see this message on your screen:

```
SELECT EMPLOYEE_ID, LAST_NAME, JO_ID, SALARY
                               *
ERROR at line 1:
ORA-00904: invalid column name
```

Examine the error message; it indicates an invalid column name in line 1 of the query. The asterisk shows the point of error—the mis-typed column JOB_ID.

Instead of re-entering the entire command, you can correct the mistake by editing the command in the buffer. The line containing the error is now the current line. Use the CHANGE command to correct the mistake. This command has three parts, separated by slashes or any other non-alphanumeric character:

- the word CHANGE or the letter C
- the sequence of characters you want to change
- the replacement sequence of characters

The CHANGE command finds the first occurrence in the current line of the character sequence to be changed and changes it to the new sequence. You do not need to use the CHANGE command to re-enter an entire line. Re-enter the line by typing the line number followed by a space and the new text and pressing Return.

Example 3–3 Correcting the Error

To change JO_ID to JOB_ID, change the line with the CHANGE command:

```
CHANGE /JO_ID/JOB_ID
```

The corrected line appears on your screen:

```
1* SELECT EMPLOYEE_ID, FIRST_NAME, JOB_ID, SALARY
```

Now that you have corrected the error, you can use the RUN command to run the command again:

```
RUN
```

SQL*Plus correctly displays the query and its result:

```
  1  SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, SALARY
  2  FROM EMP_DETAILS_VIEW
  3* WHERE JOB_ID='SA_MAN'

EMPLOYEE_ID LAST_NAME                 JOB_ID     MONTHLY SALARY
----------- ------------------------- ---------- --------------
        145 Russell                   SA_MAN            $14,000
        146 Partners                  SA_MAN            $13,500
        147 Errazuriz                 SA_MAN            $12,000
        148 Cambrault                 SA_MAN            $11,000
        149 Zlotkey                   SA_MAN            $10,500
```

Note that the column SALARY retains the format you gave it in Example 2–4. (If you have left SQL*Plus and started again since performing Example 2–4 the column has reverted to its original format.)

For information about the significance of case in a CHANGE command and on using wildcards to specify blocks of text in a CHANGE command, refer to the CHANGE command in Chapter 8.

Adding a New Line

To insert a new line after the current line, use the INPUT command.

To insert a line before line 1, enter a zero (“0”) and follow the zero with text. SQL*Plus inserts the line at the beginning of the buffer and that line becomes line 1.

```
0 SELECT EMPLOYEE_ID
```

Example 3–4 Adding a Line

Suppose you want to add a fourth line to the SQL command you modified in Example 3–3. Since line 3 is already the current line, enter INPUT (which may be abbreviated to I) and press Return.

```
INPUT
```

SQL*Plus prompts you for the new line:

```
4
```

Enter the new line. Then press Return.

```
4 ORDER BY SALARY
```

SQL*Plus prompts you again for a new line:

```
5
```

Press Return again to indicate that you will not enter any more lines, and then use RUN to verify and re-run the query.

```
  1  SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, SALARY
  2  FROM EMP_DETAILS_VIEW
  3  WHERE JOB_ID='SA_MAN'
  4* ORDER BY SALARY

EMPLOYEE_ID LAST_NAME                 JOB_ID     MONTHLY SALARY
----------- ------------------------- ---------- --------------
        149 Zlotkey                   SA_MAN            $10,500
        148 Cambrault                 SA_MAN            $11,000
        147 Errazuriz                 SA_MAN            $12,000
        146 Partners                  SA_MAN            $13,500
        145 Russell                   SA_MAN            $14,000
```

Appending Text to a Line

To add text to the end of a line in the buffer, use the APPEND command.

1. Use the LIST command (or just the line number) to list the line you want to change.
2. Enter APPEND followed by the text you want to add. If the text you want to add begins with a blank, separate the word APPEND from the first character of the text by two blanks: one to separate APPEND from the text, and one to go into the buffer with the text.

Example 3–5 Appending Text to a Line

To append a space and the clause DESC to line 4 of the current query, first list line 4:

```
LIST 4
```

```
4* ORDER BY SALARY
```

Next, enter the following command (be sure to type two spaces between APPEND and DESC):

```
APPEND  DESC
```

```
4* ORDER BY SALARY DESC
```

Type RUN to verify the query:

```
  1  SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, SALARY
  2  FROM EMP_DETAILS_VIEW
  3  WHERE JOB_ID='SA_MAN'
  4* ORDER BY SALARY DESC

EMPLOYEE_ID LAST_NAME                 JOB_ID     MONTHLY SALARY
----------- ------------------------- ---------- --------------
        145 Russell                   SA_MAN            $14,000
        146 Partners                  SA_MAN            $13,500
        147 Errazuriz                 SA_MAN            $12,000
        148 Cambrault                 SA_MAN            $11,000
        149 Zlotkey                   SA_MAN            $10,500
```

Deleting Lines

To delete lines in the buffer, use the DEL command.

1. Use the LIST command (or just the line numbers) to list the lines you want to delete.
2. Enter DEL with an optional clause.

Suppose you want to delete the current line to the last line inclusive. Use the DEL command as shown below.

```
DEL * LAST
```

DEL makes the following line of the buffer (if any) the current line.

For more information, see the DEL command in Chapter 8.

Editing Commands with a System Editor

Your computer's host operating system may have one or more text editors that you can use to create and edit host system files. Text editors perform the same general functions as the SQL*Plus editing commands, but you may find them more familiar.

You can run your host operating system's default text editor without leaving SQL*Plus by entering the EDIT command:

```
EDIT
```

EDIT loads the contents of the buffer into your system's default text editor. You can then edit the text with the text editor's commands. When you tell the text editor to save edited text and then exit, the text is loaded back into the buffer.

To load the buffer contents into a text editor other than the default, use the SQL*Plus DEFINE command to define a variable, _EDITOR, to hold the name of the editor. For example, to define the editor to be used by EDIT as EDT, enter the following command:

```
DEFINE _EDITOR = EDT
```

You can also define the editor to be used by EDIT in your user or site profile. See "Setting Up Your SQL*Plus Environment" later in this chapter and the DEFINE and EDIT commands in Chapter 8 for more information.

Saving Commands for Later Use

Through SQL*Plus, you can store one or more commands in a file called a *command file*. After you create a command file, you can retrieve, edit, and run it. Use command files to save commands for use over time, especially complex commands or PL/SQL blocks.

Storing Commands in Command Files

You can store one or more SQL commands, PL/SQL blocks, and SQL*Plus commands in command files. You create a command file within SQL*Plus in one of three ways:

- enter a command and save the contents of the buffer
- use INPUT to enter commands and then save the buffer contents
- use EDIT to create the file from scratch using a host system text editor

Because SQL*Plus commands are not stored in the buffer, you must use one of the latter two methods to save SQL*Plus commands.

Creating a Command File by Saving the Buffer Contents

To save the current SQL command or PL/SQL block for later use, enter the SAVE command. Follow the command with a file name:

```
SAVE file_name
```

SQL*Plus adds the .SQL extension to the filename to identify it as a SQL query file. If you wish to save the command or block under a name with a different file extension, type a period at the end of the filename, followed by the extension.

> **Note:** *.sql* is the file extension used by default for files saved from SQL*Plus, You can use the SQL*Plus command, SET SUFFIX *extension*, to set the file extension you want to use.

Note that within SQL*Plus, you separate the extension from the filename with a period. Your operating system may use a different character or a space to separate the filename and the extension.

Example 3–6 Saving the Current Command

First, enter LIST:

```
LIST
```

Which lists the command currently in your buffer:

```
  1  SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, SALARY
  2  FROM EMP_DETAILS_VIEW
  3  WHERE JOB_ID='SA_MAN'
```

If the query shown is not in your buffer, re-enter the query now. Next, enter the SAVE command followed by the filename EMPLINFO:

```
SAVE EMPLINFO
```

```
Created file EMPLINFO.sql
```

Verify that the command file EMPLINFO.SQL exists by entering the SQL*Plus HOST command followed by your host operating system's file listing command:

```
HOST your_host's_file_listing_command
```

You can use the same method to save a PL/SQL block currently stored in the buffer.

Creating a Command File by Using INPUT and SAVE

If you use INPUT to enter your commands, you can enter SQL*Plus commands (as well as one or more SQL commands or PL/SQL blocks) into the buffer. You must enter the SQL*Plus commands first, and the SQL command(s) or PL/SQL block(s) last—just as you would if you were entering the commands directly to the command prompt.

You can also store a set of SQL*Plus commands you plan to use with many different queries by themselves in a command file.

Example 3–7 Saving Commands Using INPUT and SAVE

Suppose you have composed a query to display a list of salespeople and their commissions. You plan to run it once a month to keep track of how well each employee is doing.

To compose and save the query using INPUT, you must first clear the buffer:

```
CLEAR BUFFER
```

Next, use INPUT to enter the command (be sure not to type a semicolon at the end of the command):

```
INPUT
```

You are then prompted to enter each line of the script. Do not enter a semicolon at the end of any statement, otherwise SQL*Plus will unsuccessfully attempt to execute the script. SQL*Plus only expects to find SQL or PL/SQL statements in the buffer.

```
COLUMN LAST_NAME HEADING 'LAST NAME'
COLUMN SALARY HEADING 'MONTHLY SALARY' FORMAT $99,999
COLUMN COMMISSION_PCT HEADING 'COMMISSION %' FORMAT 90.90
SELECT LAST_NAME, SALARY, COMMISSION_PCT
FROM EMP_DETAILS_VIEW
WHERE JOB_ID='SA_MAN'
```

The zero in the format model for the column COMMISSION_PCT tells SQL*Plus to display an initial zero for decimal values, and a zero instead of a blank when the value of COMMISSION_PCT is zero for a given row. Format models and the COLUMN command are described in more detail in Chapter 4, "Formatting Query Results" and in the *Oracle9i SQL Reference*.

Now use the SAVE command to store your query in a file called SALES with the extension SQL:

```
SAVE SALES
```

```
Created file SALES.SQL
```

Note that you do not type a semicolon at the end of the query; if you did include a semicolon, SQL*Plus would attempt to run the buffer contents. The SQL*Plus commands in the buffer would produce an error because SQL*Plus expects to find only SQL commands in the buffer. You will learn how to run a command file later in this chapter.

To input more than one SQL command, leave out the semicolons on all the SQL commands. Then, use APPEND to add a semicolon to all but the last command. (SAVE appends a slash to the end of the file automatically; this slash tells SQL*Plus to run the last command when you run the command file.)

To input more than one PL/SQL block, enter the blocks one after another without including a period or a slash on a line between blocks. Then, for each block except the last, list the last line of the block to make it current and use INPUT in the following form to insert a slash on a line by itself:

```
INPUT /
```

Creating Command Files with a System Editor

You can also create a command file with a host operating system text editor by entering EDIT followed by the name of the file, for example:

```
EDIT SALES
```

Like the SAVE command, EDIT adds the filename extension SQL to the name unless you type a period and a different extension at the end of the filename. When you save the command file with the text editor, it is saved back into the same file.

You must include a semicolon at the end of each SQL command and a period on a line by itself after each PL/SQL block in the file. (You can include multiple SQL commands and PL/SQL blocks.)

When you create a command file using EDIT, you can also include SQL*Plus commands at the end of the file. You cannot do this when you create a command file using the SAVE command because SAVE appends a slash to the end of the file. This slash would cause SQL*Plus to run the command file twice, once upon reaching the semicolon at the end of the last SQL command (or the slash after the last PL/SQL block) and once upon reaching the slash at the end of the file.

Placing Comments in Command Files

You can enter comments in a command file in three ways:

- using the SQL*Plus REMARK command for single line comments.
- using the SQL comment delimiters /*... */ for single of multi line comments.
- using ANSI/ISO (American National Standards Institute/International Standards Organization) comments -- for single line comments.

  For further information about using comments in command files, see "Notes on Placing Comments" later in this chapter.

Using the REMARK Command

Use the REMARK command on a line by itself in a command file, followed by comments on the same line. To continue the comments on additional lines, enter additional REMARK commands. Do not place a REMARK command between different lines of a single SQL command.

```
REMARK Commission Report;
REMARK to be run monthly.;
COLUMN LAST_NAME HEADING 'LAST_NAME';
COLUMN SALARY HEADING 'MONTHLY SALARY' FORMAT $99,999;
COLUMN COMMISSION_PCT HEADING 'COMMISSION %' FORMAT 90.90;
REMARK Includes only salesmen;
SELECT LAST_NAME, SALARY, COMMISSION_PCT
FROM EMP_DETAILS_VIEW
WHERE JOB_ID='SA_MAN'
```

Using /*...*/

Enter the SQL comment delimiters, /*...*/, on separate lines in your command file, on the same line as a SQL command, or on a line in a PL/SQL block.

You must enter a space after the slash-asterisk(/*) beginning a comment, otherwise the comment is treated as a command, and the slash is interpreted as an execute command, executing any command in the SQL*Plus buffer.

The comments can span multiple lines, but cannot be nested within one another:

```
/* Commission Report
 to be run monthly. */
COLUMN LAST_NAME HEADING 'LAST_NAME';
COLUMN SALARY HEADING 'MONTHLY SALARY' FORMAT $99,999;
COLUMN COMMISSION_PCT HEADING 'COMMISSION %' FORMAT 90.90;
REMARK Includes only salesmen;
SELECT LAST_NAME, SALARY, COMMISSION_PCT
FROM EMP_DETAILS_VIEW
/* Include only salesmen.*/
WHERE JOB_ID='SA_MAN'
```

If you enter a SQL comment directly at the command prompt, SQL*Plus does not store the comment in the buffer.

Using --

You can use ANSI/ISO "--" style comments within SQL statements, PL/SQL blocks, or SQL*Plus commands. Since there is no ending delimiter, the comment cannot span multiple lines.

For PL/SQL and SQL, enter the comment after a command on a line, or on a line by itself:

```
-- Commissions report to be run monthly
DECLARE --block for reporting monthly sales
```

For SQL*Plus commands, you can only include "--" style comments if they are on a line by themselves. For example, these comments are legal:

```
-- set maximum width for LONG to 777
SET LONG 777
```

This comment is illegal:

```
SET LONG 777 -- set maximum width for LONG to 777
```

If you enter the following SQL*Plus command, SQL*Plus interprets it as a comment and does not execute the command:

```
    -- SET LONG 777
```

Notes on Placing Comments

SQL*Plus generally does not parse or execute input it identifies as a comment.

SQL*Plus does not have a SQL or PL/SQL command parser. It scans the first few keywords of each new statement to determine the command type, SQL, PL/SQL or SQL*Plus. Comments in some locations can prevent SQL*Plus from correctly identifying the command type, giving unexpected results. The following usage notes may help you to use SQL*Plus comments more effectively:

1. Do not put comments within the first few keywords of a statement. For example:

```
SQL> CREATE OR REPLACE
  2  /* HELLO */
  3  PROCEDURE HELLO AS
  4  BEGIN
  5  DBMS_OUTPUT.PUT_LINE('HELLO');

Warning: Procedure created with compilation errors.
```

   The location of the comment prevents SQL*Plus from recognizing the command as a PL/SQL command. SQL*Plus submits the block to the server when it sees the slash "/" at the beginning of the comment, which it interprets as the "/" statement terminator. Move the comment to avoid this error. For example:

```
 CREATE OR REPLACE PROCEDURE
  2  /* HELLO */
  3  HELLO AS
  4  BEGIN
  5  DBMS_OUTPUT.PUT_LINE('HELLO');
  6  END;
  7  /

Procedure created.
```

2. Do not put comments after statement terminators (period, semicolon or slash). For example, if you enter:

```
SELECT 'Y' FROM DUAL; -- TESTING
```

   You get the following error:

```
SELECT 'Y' FROM DUAL; -- TESTING
                     *
ERROR at line 1:
ORA-00911: invalid character
```

SQL*Plus expects no text after statement terminators on the same line and is unable to recognize the comment.

3. Do not put statement termination characters at the end of a comment line or after comments in a SQL statement or a PL/SQL block. For example, if you enter:

```
SELECT *
-- COMMENT;
```

You get the following error:

```
-- COMMENT
         *
ERROR at line 2:
ORA-00923: FROM keyword not found where expected
```

The semicolon is interpreted as a statement terminator and SQL*Plus submits the partially formed SQL command to the server for processing, resulting in an error.

4. Do not use ampersand characters '&' in comments in a SQL statement or PL/SQL block. For example, if you enter a script such as:

```
SELECT REGION_NAME, CITY
/* THIS & THAT */
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000;
```

It prompts for the value of &that:

```
Enter value for that:
old   2: /* THIS & THAT */
new   2: /* THIS  */

REGION_NAME               CITY
------------------------- ------------------------------
Americas                  Seattle
Americas                  Seattle
Americas                  Seattle
Europe                    Oxford
Europe                    Oxford
Americas                  Toronto

6 rows selected.
```

SQL*Plus interprets text after the ampersand character "&" as a substitution variable and prompts for the value of the variable. You can SET DEFINE OFF to prevent scanning for the substitution character.

For more information on substitution and termination characters, see DEFINE, SQLTERMINATOR and SQLBLANKLINES in the SET command in Chapter 8.

Retrieving Command Files

If you want to place the contents of a command file in the buffer, you must retrieve the command from the file in which it is stored. You can retrieve a command file using the SQL*Plus command GET.

Just as you can save a query from the buffer to a file with the SAVE command, you can retrieve a query from a file to the buffer with the GET command:

```
GET file_name
```

When appropriate to the operating system, SQL*Plus adds a period and the extension SQL to the filename unless you type a period at the end of the filename followed by a different extension. For information about setting the file suffix, see SET SUFFIX in Chapter 8, "Command Reference".

Example 3–8 Retrieving a Command File

Suppose you need to retrieve the SALES file in a later session. You can retrieve the file by entering the GET command. To retrieve the file SALES, enter

```
GET SALES
```

SQL*Plus retrieves the contents of the file SALES.SQL into the SQL buffer and lists it on the screen:

```
  1  COLUMN LAST_NAME HEADING 'LAST NAME'
  2  COLUMN SALARY HEADING 'MONTHLY SALARY' FORMAT $99,999
  3  COLUMN COMMISSION_PCT HEADING 'COMMISSION %' FORMAT 90.90
  4  SELECT LAST_NAME, SALARY, COMMISSION_PCT
  5  FROM EMP_DETAILS_VIEW
  6* WHERE JOB_ID='SA_MAN'
```

The file should contain a single SQL statement or PL/SQL block. SQL*Plus commands or multiple statements or blocks will be loaded, but will give errors if run with "/" or RUN.

Running Command Files

The START command retrieves a command file and runs the command(s) it contains. Use START to run a command file containing SQL commands, PL/SQL blocks, and SQL*Plus commands. You can have many commands in the file. Follow the START command with the name of the file:

```
START file_name
```

If the file has the extension SQL, you need not add the period and the extension SQL to the filename.

Example 3–9 Running a Command File

To retrieve and run the command stored in SALES.SQL, enter

```
START SALES
```

SQL*Plus runs the commands in the file SALES and displays the results of the commands on your screen, formatting the query results according to the SQL*Plus commands in the file:

```
LAST NAME                 MONTHLY SALARY COMMISSION %
------------------------- -------------- ------------
Russell                          $14,000         0.40
Partners                         $13,500         0.30
Errazuriz                        $12,000         0.30
Cambrault                        $11,000         0.30
Zlotkey                          $10,500         0.20
```

To see the commands as SQL*Plus “enters” them, you can set the ECHO variable of the SET command to ON. The ECHO variable controls the listing of the commands in command files run with the START, @ and @@ commands. Setting the ECHO variable to OFF suppresses the listing.

You can also use the @ (“at” sign) command to run a command file:

```
@SALES
```

The @ command lists and runs the commands in the specified command file in the same manner as START. SET ECHO affects the @ command as it affects the START command.

START, @ and @@ leave the last SQL command or PL/SQL block in the command file in the buffer.

Running a Command File as You Start SQL*Plus

To run a command file as you start SQL*Plus, use one of the following four options:

- Follow the SQLPLUS command with your username, a slash, your password, a space, @, and the name of the file:

```
SQLPLUS HR/HR @SALES
```

  SQL*Plus starts and runs the command file.

- Follow the SQLPLUS command and your username with a space, @, and the name of the file:

```
SQLPLUS HR @SALES
```

  SQL*Plus prompts you for your password, starts, and runs the command file.

- Include your username as the first line of the file. Follow the SQLPLUS command with @ and the filename. SQL*Plus prompts for your password, starts, and runs the file.
- Include your username, a slash (/), and your password as the first line of the file. Follow the SQLPLUS command with @ and the filename. SQL*Plus starts and runs the file. Please consider the security risks of exposing your password in the file before using this technique.

Nesting Command Files

To run a series of command files in sequence, first create a command file containing several START commands, each followed by the name of a command file in the sequence. Then run the command file containing the START commands. For example, you could include the following START commands in a command file named SALESRPT:

```
START Q1SALES
START Q2SALES
START Q3SALES
START Q4SALES
START YRENDSLS
```

Note: The @@ command may be useful in this example. See the @@ (double "at" sign) command in Chapter 8 for more information.

Modifying Command Files

You can modify an existing command file in two ways:

- using the EDIT command
- using GET, the SQL*Plus editing commands, and SAVE

To edit an existing command file with the EDIT command, follow the word EDIT with the name of the file.

For example, to edit an existing file named PROFIT that has the extension SQL, enter the following command:

```
EDIT PROFIT
```

Remember that EDIT assumes the file extension SQL if you do not specify one.

To edit an existing file using GET, the SQL*Plus editing commands, and SAVE, first retrieve the file with GET, then edit the file with the SQL*Plus editing commands, and finally save the file with the SAVE command.

Note that if you want to replace the contents of an existing command file with the command or block in the buffer, you must use the SAVE command and follow the filename with the word REPLACE.

For example:

```
GET MYREPORT
```

```
1* SELECT * FROM EMP
```

```
CHANGE/EMP/EMP_DETAILS_VIEW
```

```
1* SELECT * FROM EMP_DETAILS_VIEW
```

```
SAVE MYREPORT REPLACE
```

```
Wrote file MYREPORT
```

If you want to append the contents of the buffer to the end of an existing command file, use the SAVE command and follow the filename with the word APPEND:

```
SAVE file_name APPEND
```

Exiting from a Command File with a Return Code

If your command file generates a SQL error while running from a batch file on the host operating system, you may want to abort the command file and exit with a return code. Use the SQL*Plus command WHENEVER SQLERROR to do this; see the WHENEVER SQLERROR command in Chapter 8 for more information.

Similarly, the WHENEVER OSERROR command may be used to exit if an operating system error occurs. See the WHENEVER OSERROR command in Chapter 8 for more information.

Setting Up Your SQL*Plus Environment

You may wish to set up your SQL*Plus environment in a particular way (such as showing the current time as part of the SQL*Plus command prompt) and then reuse those settings with each session. You can do this through a host operating system file called LOGIN with the file extension SQL (also called your *User Profile*). The exact name of this file is system dependent; see the Oracle installation and user's guide provided for your operating system for the precise name.

You can add any SQL commands, PL/SQL blocks, or SQL*Plus commands to this file; when you start SQL*Plus, it automatically searches for your LOGIN file (first in your local directory and then on a system-dependent path) and runs the commands it finds there. (You may also have a Site Profile, for example, GLOGIN.SQL which is run before LOGIN.SQL. See"Setting Up the Site Profile" for more information on the relationship of Site and User Profiles.)

Modifying Your LOGIN File

You can modify your LOGIN file just as you would any other command file. You may wish to add some of the following commands to the LOGIN file:

`SET LINESIZE`	Followed by a number, sets the number of characters as page width of the query results.
`SET NUMFORMAT`	Followed by a number format (such as $99,999), sets the default format for displaying numbers in query results.
`SET PAGESIZE`	Followed by a number, sets the number of lines per page.
`SET PAUSE`	Followed by ON, causes SQL*Plus to pause at the beginning of each page of output (SQL*Plus continues scrolling after you enter Return). Followed by text, sets the text to be displayed each time SQL*Plus pauses (you must also set PAUSE to ON).

`SET TIME`	Followed by ON, displays the current time before each command prompt.

See the SET command in Chapter 8 for more information on these and other SET command variables you may wish to set in your SQL*Plus LOGIN file.

Storing and Restoring SQL*Plus System Variables

You can store the current SQL*Plus system ("SET") variables in a host operating system file (a command file) with the STORE command. If you alter any variables, this command file can be run to restore the original values. This is useful if you want to reset system variables after running a report that alters them.

To store the current setting of all system variables, enter

```
STORE SET file_name
```

By default, SQL*Plus adds the extension "SQL" to the file name. If you want to use a different file extension, type a period at the end of the file name, followed by the extension. Alternatively, you can use the SET SUFFIX command to change the default file extension.

Restoring the System Variables

To restore the stored system variables, enter

```
START file_name
```

If the file has the default extension (as specified by the SET SUFFIX command), you do not need to add the period and extension to the file name.

You can also use the @ ("at" sign) or the @@ (double "at" sign) commands to run the command file.

Example 3–10 Storing and Restoring SQL*Plus System Variables

To store the current values of the SQL*Plus system variables in a new command file "plusenv.sql":

```
STORE SET plusenv
```

```
Created file plusenv
```

Now the value of any system variable can be changed:

```
SHOW PAGESIZE
```

```
PAGESIZE 24
```

```
SET PAGESIZE 60
SHOW PAGESIZE
```

```
PAGESIZE 60
```

The original values of system variables can then be restored from the command file:

```
START plusenv
SHOW PAGESIZE
```

```
PAGESIZE 24
```

Writing Interactive Commands

The following features of SQL*Plus make it possible for you to set up command files that allow end-user input:

- defining user variables
- substituting values in commands
- using the START command to provide values
- prompting for values

Defining User Variables

You can define variables, called *user variables*, for repeated use in a single command file by using the SQL*Plus DEFINE command. Note that you can also define user variables to use in titles and to save your keystrokes (by defining a long string as the value for a variable with a short name).

Example 3–11 Defining a User Variable

To define a user variable L_NAME and give it the value “SMITH”, enter the following command:

```
DEFINE L_NAME = SMITH
```

To confirm the variable definition, enter DEFINE followed by the variable name:

```
DEFINE L_NAME
```

```
DEFINE L_NAME = "SMITH" (CHAR)
```

To list all user variable definitions, enter DEFINE by itself at the command prompt. Note that any user variable you define explicitly through DEFINE takes only CHAR values (that is, the value you assign to the variable is always treated as a CHAR datatype). You can define a user variable of datatype NUMBER implicitly through the ACCEPT command. You will learn more about the ACCEPT command later in this chapter.

To delete a user variable, use the SQL*Plus command UNDEFINE followed by the variable name.

Using Substitution Variables

Suppose you want to write a query like the one in SALES (see Example 3–7) to list the employees with various jobs, not just those whose job is SA_MAN. You could do that by editing a different CHAR value into the WHERE clause each time you run the command, but there is an easier way.

By using a *substitution variable* in place of the value SA_MAN in the WHERE clause, you can get the same results you would get if you had written the values into the command itself.

A substitution variable is a user variable name preceded by one or two ampersands (&). When SQL*Plus encounters a substitution variable in a command, SQL*Plus executes the command as though it contained the value of the substitution variable, rather than the variable itself.

For example, if the variable SORTCOL has the value JOB_ID and the variable MYTABLE has the value EMP_DETAILS_VIEW, SQL*Plus executes the commands

```
SELECT &SORTCOL, SALARY
FROM &MYTABLE
WHERE SALARY>12000;
```

as if they were

```
SELECT JOB_ID, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000;
```

Where and How to Use Substitution Variables

You can use substitution variables anywhere in SQL and SQL*Plus commands, except as the first word entered at the command prompt. When SQL*Plus encounters an undefined substitution variable in a command, SQL*Plus prompts you for the value.

You can enter any string at the prompt, even one containing blanks and punctuation. If the SQL command containing the reference should have quote marks around the variable and you do not include them there, the user must include the quotes when prompted.

SQL*Plus reads your response from the keyboard, even if you have redirected terminal input or output to a file. If a terminal is not available (if, for example, you run the command file in batch mode), SQL*Plus uses the redirected file.

After you enter a value at the prompt, SQL*Plus lists the line containing the substitution variable twice: once before substituting the value you enter and once after substitution. You can suppress this listing by setting the SET command variable VERIFY to OFF.

You should avoid creating substitution variables with names that may be identical to values that you will pass to them, as unexpected results can occur. If a value supplied for a substitution variable matches a variable name, then the contents of the matching variable are used instead of the supplied value.

Example 3–12 Using Substitution Variables

Create a command file named STATS, to be used to calculate a subgroup statistic (the maximum value) on a numeric column:

```
CLEAR BUFFER
INPUT
    SELECT &GROUP_COL, MAX(&NUMBER_COL) MAXIMUM
    FROM &TABLE
    GROUP BY &GROUP_COL
    .
SAVE STATS
```

```
Created file STATS
```

Now run the command file STATS:

```
@STATS
```

And respond to the prompts for values as shown:

```
Enter value for group_col: JOB_ID
old   1: SELECT   &GROUP_COL,
new   1: SELECT   JOB_ID,
Enter value for number_col: SALARY
old   2:          MAX(&NUMBER_COL) MAXIMUM
new   2:          MAX(SALARY) MAXIMUM
```

```
Enter value for table: EMP_DETAILS_VIEW
old   3: FROM     &TABLE
new   3: FROM     EMP_DETAILS_VIEW
Enter value for group_col: JOB_ID
old   4: GROUP BY &GROUP_COL
new   4: GROUP BY JOB_ID
```

SQL*Plus displays the following output:

```
JOB_ID        MAXIMUM
---------- ----------
AC_ACCOUNT       8300
AC_MGR          12000
AD_ASST          4400
AD_PRES         24000
AD_VP           17000
FI_ACCOUNT       9000
FI_MGR          12000
HR_REP           6500
IT_PROG          9000
MK_MAN          13000
MK_REP           6000

JOB_ID        MAXIMUM
---------- ----------
PR_REP          10000
PU_CLERK         3100
PU_MAN          11000
SA_MAN          14000
SA_REP          11500
SH_CLERK         4200
ST_CLERK         3600
ST_MAN           8200

19 rows selected.
```

If you wish to append characters immediately after a substitution variable, use a period to separate the variable from the character. For example:

```
SELECT SALARY FROM EMP_DETAILS_VIEW WHERE EMPLOYEE_ID='&X.5';
Enter value for X:  20
```

is interpreted as

```
SELECT SALARY FROM EMP_DETAILS_VIEW WHERE EMPLOYEE_ID='205';
```

Avoiding Unnecessary Prompts for Values

Suppose you wanted to expand the file STATS to include the minimum, sum, and average of the "number" column. You may have noticed that SQL*Plus prompted you twice for the value of GROUP_COL and once for the value of NUMBER_COL in Example 3–12, and that each GROUP_COL or NUMBER_COL had a single ampersand in front of it. If you were to add three more functions—using a single ampersand before each—to the command file, SQL*Plus would prompt you a total of four times for the value of the number column.

You can avoid being re-prompted for the group and number columns by adding a second ampersand in front of each GROUP_COL and NUMBER_COL in STATS. SQL*Plus automatically DEFINEs any substitution variable preceded by two ampersands, but does not DEFINE those preceded by only one ampersand. When you have DEFINEd a variable, 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*.

Example 3–13 Using Double Ampersands

To expand the command file STATS using double ampersands and then run the file, first suppress the display of each line before and after substitution:

```
SET VERIFY OFF
```

Now retrieve and edit STATS by entering the following commands:

```
GET STATS
SELECT   &GROUP_COL,
MAX(&NUMBER_COL) MAXIMUM
FROM     &TABLE
GROUP BY &GROUP_COL
2
  2* MAX(&NUMBER_COL) MAXIMUM
APPEND ,
  2* MAX(&NUMBER_COL) MAXIMUM,
CHANGE/&/&&
  2* MAX(&&NUMBER_COL) MAXIMUM,
I
  3i MIN (&&NUMBER_COL) MINIMUM,
  4i SUM(&&NUMBER_COL)  TOTAL,
  5i AVG(&&NUMBER_COL)  AVERAGE
  6i
  1* SELECT   &GROUP_COL,
```

```
CHANGE/&/&&
  1* SELECT   &&GROUP_COL,
7
  7* GROUP BY &GROUP_COL
CHANGE/&/&&/
  7* GROUP BY &&GROUP_COL
SAVE STATS2
Created file STATS2
```

Finally, run the command file STATS2 and respond to the prompts as follows:

```
START STATS2
Enter value for group_col: JOB_ID
Enter value for number_col: SALARY
Enter value for table: EMP_DETAILS_VIEW
```

SQL*Plus displays the following output:

```
JOB_ID        MAXIMUM    MINIMUM      TOTAL    AVERAGE
---------- ---------- ---------- ---------- ----------
AC_ACCOUNT       8300       8300       8300       8300
AC_MGR          12000      12000      12000      12000
AD_ASST          4400       4400       4400       4400
AD_PRES         24000      24000      24000      24000
AD_VP           17000      17000      34000      17000
FI_ACCOUNT       9000       6900      39600       7920
FI_MGR          12000      12000      12000      12000
HR_REP           6500       6500       6500       6500
IT_PROG          9000       4200      28800       5760
MK_MAN          13000      13000      13000      13000
MK_REP           6000       6000       6000       6000

JOB_ID        MAXIMUM    MINIMUM      TOTAL    AVERAGE
---------- ---------- ---------- ---------- ----------
PR_REP          10000      10000      10000      10000
PU_CLERK         3100       2500      13900       2780
PU_MAN          11000      11000      11000      11000
SA_MAN          14000      10500      61000      12200
SA_REP          11500       6100     250500       8350
SH_CLERK         4200       2500      64300       3215
ST_CLERK         3600       2100      55700       2785
ST_MAN           8200       5800      36400       7280

19 rows selected.
```

Note that you were prompted for the values of NUMBER_COL and GROUP_COL only once. If you were to run STATS2 again during the current session, you would be prompted for TABLE (because its name has a single ampersand and the variable is therefore not DEFINEd) but not for GROUP_COL or NUMBER_COL (because their names have double ampersands and the variables are therefore DEFINEd).

Before continuing, set the system variable VERIFY back to ON:

```
SET VERIFY ON
```

Restrictions

You cannot use substitution variables in the buffer editing commands, APPEND, CHANGE, DEL, and INPUT, nor in other commands where substitution would be meaningless, such as in SQL*Plus comments (REMARK, /*... */ or --). The buffer editing commands, APPEND, CHANGE, and INPUT, treat text beginning with "&" or "&&" literally, as any other text string.

System Variables

The following system variables, specified with the SQL*Plus SET command, affect substitution variables:

`SET DEFINE`	Defines the substitution character (by default the ampersand "&") and turns substitution on and off.
`SET ESCAPE`	Defines an escape character you can use before the substitution character. The escape character instructs SQL*Plus to treat the substitution character as an ordinary character rather than as a request for variable substitution. The default escape character is a backslash (\).
`SET VERIFY ON`	Lists each line of the command file before and after substitution.
`SET CONCAT`	Defines the character that separates the name of a substitution variable or parameter from characters that immediately follow the variable or parameter—by default the period (.).

For more information about system variables, see the SET command in the "Command Reference" in Chapter 8.

Passing Parameters through the START Command

You can bypass the prompts for values associated with substitution variables by passing values to parameters in a command file through the START command.

You do this by placing an ampersand (&) followed by a numeral in the command file in place of a substitution variable. Each time you run this command file, START replaces each &1 in the file with the first value (called an argument) after START *filename*, then replaces each &2 with the second value, and so forth.

For example, you could include the following commands in a command file called MYFILE:

```
SELECT * FROM EMP_DETAILS_VIEW
WHERE JOB_ID='&1'
AND SALARY='&2';
```

In the following START command, SQL*Plus would substitute CLERK for &1 and 7900 for &2 in the command file MYFILE:

```
START MYFILE PU_CLERK 3100
```

When you use arguments with the START command, SQL*Plus DEFINEs each parameter in the command file with the value of the appropriate argument.

Example 3–14 Passing Parameters through START

To create a new command file based on SALES that takes a parameter specifying the job to be displayed, enter

```
GET SALES
  1  COLUMN LAST_NAME HEADING 'LAST NAME'
  2  COLUMN SALARY  HEADING 'MONTHLY SALARY' FORMAT $99,999
  3  COLUMN COMMISSION_PCT HEADING 'COMMISSION %' FORMAT 90.90
  4  SELECT LAST_NAME, SALARY, COMMISSION_PCT
  5  FROM EMP_DETAILS_VIEW
  6* WHERE JOB_ID='SA_MAN'
6
  6* WHERE JOB_ID='SA_MAN'
CHANGE /SA_MAN/&1
  6* WHERE JOB_ID='&1'
SAVE ONEJOB
Created file ONEJOB
```

Now run the command with the parameter CLERK:

```
START ONEJOB SA_MAN
```

SQL*Plus lists the line of the SQL command that contains the parameter, before and after replacing the parameter with its value, and then displays the output:

```
old   3: WHERE JOB_ID='&1'
new   3: WHERE JOB_ID='SA_MAN'

LAST NAME                 MONTHLY SALARY COMMISSION %
------------------------- -------------- ------------
Russell                          $14,000         0.40
Partners                         $13,500         0.30
Errazuriz                        $12,000         0.30
Cambrault                        $11,000         0.30
Zlotkey                          $10,500         0.20
```

You can use any number of parameters in a command file. Within a command file, you can refer to each parameter any number of times, and can include the parameters in any order.

> **Note:** You cannot use parameters when you run a command with RUN or slash (/). You must store the command in a command file and run it with START or @.

Before continuing, return the columns to their original heading by entering the following command:

```
CLEAR COLUMN
```

Communicating with the User

Three SQL*Plus commands—PROMPT, ACCEPT, and PAUSE—help you communicate with the end user. These commands enable you to send messages to the screen and receive input from the user, including a simple Return. You can also use PROMPT and ACCEPT to customize the prompts for values SQL*Plus automatically generates for substitution variables.

Prompting for and Accepting User Variable

Through PROMPT and ACCEPT, you can send messages to the end user and accept values as end-user input. PROMPT displays a message you specify on-screen; use it to give directions or information to the user. ACCEPT prompts the user for a value and stores it in the user variable you specify. Use PROMPT in conjunction with ACCEPT when your prompt for the value spans more than one line.

Example 3–15 Prompting for and Accepting Input

To direct the user to supply a report title and to store the input in the variable MYTITLE for use in a subsequent query, first clear the buffer:

```
CLEAR BUFFER
```

Next, set up a command file as shown and save this file as PROMPT1:

```
INPUT
```

```
  4  PROMPT Enter a title of up to 30 characters
  5  ACCEPT MYTITLE PROMPT 'Title: '
  6  TTITLE LEFT MYTITLE SKIP 2
  7  SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY
  8  FROM EMP_DETAILS_VIEW
  9  WHERE JOB_ID='SA_MAN'
  10
```

```
SAVE PROMPT1
```

```
Created file PROMPT1.sql
```

The TTITLE command sets the top title for your report. For more information about the TTITILE command, see "Defining Page and Report Titles and Dimensions" in Chapter 4.

Finally, run the command file, responding to the prompt for the title as shown:

```
START PROMPT1

Enter a title of up to 30 characters
Title: Department Report
Department Report

EMPLOYEE_ID FIRST_NAME           LAST_NAME                     SALARY
----------- -------------------- ------------------------- ----------
        145 John                 Russell                        14000
        146 Karen                Partners                       13500
        147 Alberto              Errazuriz                      12000
        148 Gerald               Cambrault                      11000
        149 Eleni                Zlotkey                        10500
```

Before continuing, turn the TTITLE command off:

```
TTITLE OFF
```

Customizing Prompts for Substitution Variable

If you want to customize the prompt for a substitution variable value, use PROMPT and ACCEPT in conjunction with the substitution variable, as shown in the following example.

Example 3–16 Using PROMPT and ACCEPT in Conjunction with Substitution Variables

As you have seen in Example 3–15, SQL*Plus automatically generates a prompt for a value when you use a substitution variable. You can replace this prompt by including PROMPT and ACCEPT in the command file with the query that references the substitution variable. First clear the buffer with:

```
CLEAR BUFFER
```

To create such a file, enter the following:

```
INPUT
    PROMPT Enter a valid employee ID
    PROMPT For Example 145, 206
    ACCEPT ENUMBER NUMBER PROMPT 'Employee ID. :'
    SELECT FIRST_NAME, LAST_NAME, SALARY
    FROM EMP_DETAILS_VIEW
    WHERE EMPLOYEE_ID=&ENUMBER
    .
```

Save this file as PROMPT2. Next, run this command file. SQL*Plus prompts for the value of ENUMBER using the text you specified with PROMPT and ACCEPT:

```
START PROMPT2
```

SQL*Plus prompts you to enter an Employee ID:

```
Enter a valid employee ID
For Example 145, 206

Employee ID. :205
```

```
old   3: WHERE EMPLOYEE_ID=&ENUMBER
new   3: WHERE EMPLOYEE_ID=       205

Department Report

FIRST_NAME           LAST_NAME                     SALARY
-------------------- ------------------------- ----------
Shelley              Higgins                        12000
```

What would happen if you typed characters instead of numbers? Since you specified NUMBER after the variable name in the ACCEPT command, SQL*Plus will not accept a non-numeric value:

Try entering characters instead of numbers to the prompt for “Employee. ID.”, SQL*Plus will respond with an error message and prompt you again to re-enter the correct number:

```
START PROMPT2
```

When SQL*Plus prompts you to enter an Employee ID, enter the word "one" instead of a number:

```
Enter a valid employee ID
For Example 145, 206

Employee ID. :one

SP2-0425: "one" is not a valid number
```

Sending a Message and Accepting Return as Input

If you want to display a message on the user’s screen and then have the user enter Return after reading the message, use the SQL*Plus command PAUSE. For example, you might include the following lines in a command file:

```
PROMPT Before continuing, make sure you have your account card.
PAUSE Press RETURN to continue.
```

Clearing the Screen

If you want to clear the screen before displaying a report (or at any other time), include the SQL*Plus CLEAR command with its SCREEN clause at the appropriate point in your command file, using the following format:

```
CLEAR SCREEN
```

Before continuing to the next section, reset all columns to their original formats and headings by entering the following command:

```
CLEAR COLUMNS
```

Using Bind Variables

Suppose that you want to be able to display the variables you use in your PL/SQL subprograms in SQL*Plus or use the same variables in multiple subprograms. If you

declare a variable in a PL/SQL subprogram, you cannot display that variable in SQL*Plus. Use a bind variable in PL/SQL to access the variable from SQL*Plus.

Bind variables are variables you create in SQL*Plus and then reference in PL/SQL or SQL. If you create a bind variable in SQL*Plus, you can use the variable as you would a declared variable in your PL/SQL subprogram and then access the variable from SQL*Plus. You can use bind variables for such things as storing return codes or debugging your PL/SQL subprograms.

Because bind variables are recognized by SQL*Plus, you can display their values in SQL*Plus or reference them in PL/SQL subprograms that you run in SQL*Plus.

Creating Bind Variables

You create bind variables in SQL*Plus with the VARIABLE command. For example

```
VARIABLE ret_val NUMBER
```

This command creates a bind variable named `ret_val` with a datatype of NUMBER. For more information, see the VARIABLE command in Chapter 8. (To list all bind variables created in a session, type VARIABLE without any arguments.)

Referencing Bind Variables

You reference bind variables in PL/SQL by typing a colon (:) followed immediately by the name of the variable. For example

```
:ret_val := 1;
```

To change this bind variable in SQL*Plus, you must enter a PL/SQL block. For example:

```
VARIABLE ret_val NUMBER
BEGIN
 :ret_val:=4;
END;
/
```

```
PL/SQL procedure successfully completed.
```

This command assigns a value to the bind variable named `ret_val`.

Displaying Bind Variables

To display the value of a bind variable in SQL*Plus, you use the SQL*Plus PRINT command. For example:

```
PRINT RET_VAL
```

```
   RET_VAL
----------
         4
```

This command displays a bind variable named `ret_val`. For more information about displaying bind variables, see the PRINT command in the "Command Reference" in Chapter 8.

Using REFCURSOR Bind Variables

SQL*Plus REFCURSOR bind variables allow SQL*Plus to fetch and format the results of a SELECT statement contained in a PL/SQL block.

REFCURSOR bind variables can also be used to reference PL/SQL cursor variables in stored procedures. This allows you to store SELECT statements in the database and reference them from SQL*Plus.

A REFCURSOR bind variable can also be returned from a stored function.

Note: You must have Oracle7, Release 7.3 or above to assign the return value of a stored function to a REFCURSOR variable.

Example 3–17 Creating, Referencing, and Displaying REFCURSOR Bind Variables

To create, reference and display a REFCURSOR bind variable, first declare a local bind variable of the REFCURSOR datatype

```
VARIABLE employee_info REFCURSOR
```

Next, enter a PL/SQL block that uses the bind variable in an OPEN... FOR SELECT statement. This statement opens a cursor variable and executes a query. See the *PL/SQL User's Guide and Reference* for information on the OPEN command and cursor variables.

In this example we are binding the SQL*Plus *employee_info* bind variable to the cursor variable.

```
BEGIN
 OPEN :employee_info FOR SELECT EMPLOYEE_ID, SALARY FROM EMP_DETAILS_VIEW WHERE
JOB_ID='SA_MAN' ;
END;
 /
```

```
PL/SQL procedure successfully completed.
```

The results from the SELECT statement can now be displayed in SQL*Plus with the PRINT command.

```
PRINT employee_info
```

```
EMPLOYEE_ID     SALARY
----------- ----------
        145      14000
        146      13500
        147      12000
        148      11000
        149      10500
```

The PRINT statement also closes the cursor. To reprint the results, the PL/SQL block must be executed again before using PRINT.

Example 3–18 Using REFCURSOR Variables in Stored Procedures

A REFCURSOR bind variable is passed as a parameter to a procedure. The parameter has a REF CURSOR type. First, define the type.

```
CREATE OR REPLACE PACKAGE cv_types AS
TYPE EmpInfoTyp is REF CURSOR RETURN emp%ROWTYPE;
END cv_types;
/
```

```
Package created.
```

Next, create the stored procedure containing an OPEN... FOR SELECT statement.

```
CREATE OR REPLACE PROCEDURE EmpInfo_rpt
(emp_cv IN OUT cv_types.EmpInfoTyp) AS
BEGIN
OPEN emp_cv FOR SELECT EMPLOYEE_ID, SALARY FROM EMP_DETAILS_VIEW -
WHERE JOB_ID='SA_MAN' ;
 END;
 /
```

```
Procedure created.
```

Execute the procedure with a SQL*Plus bind variable as the parameter.

```
VARIABLE odcv REFCURSOR
EXECUTE EmpInfo_rpt(:odcv)
```

```
PL/SQL procedure successfully completed.
```

Now print the bind variable.

```
PRINT odcv
```

```
EMPLOYEE_ID     SALARY
----------- ----------
        145      14000
        146      13500
        147      12000
        148      11000
        149      10500
```

The procedure can be executed multiple times using the same or a different REFCURSOR bind variable.

```
VARIABLE pcv REFCURSOR
EXECUTE EmpInfo_rpt(:pcv)
```

```
PL/SQL procedure successfully completed.
```

```
PRINT pcv
```

```
EMPLOYEE_ID     SALARY
----------- ----------
        145      14000
        146      13500
        147      12000
        148      11000
        149      10500
```

Example 3–19 Using REFCURSOR Variables in Stored Functions

Create a stored function containing an OPEN... FOR SELECT statement:

```
CREATE OR REPLACE FUNCTION EmpInfo_fn RETURN -
cv_types.EmpInfo IS
resultset cv_types.EmpInfoTyp;
BEGIN
OPEN resultset FOR SELECT EMPLOYEE_ID, SALARY FROM EMP_DETAILS_VIEW -
WHERE JOB_ID='SA_MAN' ;
```

```
RETURN(resultset);
END;
 /
Function created.
```

Execute the function.

```
VARIABLE rc REFCURSOR
EXECUTE :rc := EmpInfo_fn

PL/SQL procedure successfully completed.
```

Now print the bind variable.

```
PRINT rc

EMPLOYEE_ID     SALARY
----------- ----------
        145      14000
        146      13500
        147      12000
        148      11000
        149      10500
```

The function can be executed multiple times using the same or a different REFCURSOR bind variable.

```
EXECUTE :rc := EmpInfo_fn

PL/SQL procedure successfully completed.

PRINT rc

EMPLOYEE_ID     SALARY
----------- ----------
        145      14000
        146      13500
        147      12000
        148      11000
        149      10500
```

Tracing Statements

You can automatically get a report on the execution path used by the SQL optimizer and the statement execution statistics. The report is generated after successful SQL DML (that is, SELECT, DELETE, UPDATE and INSERT) statements. It is useful for monitoring and tuning the performance of these statements.

Controlling the Report

You can control the report by setting the AUTOTRACE system variable.

`SET AUTOTRACE OFF`	No AUTOTRACE report is generated. This is the default.
`SET AUTOTRACE ON EXPLAIN`	The AUTOTRACE report shows only the optimizer execution path.
`SET AUTOTRACE ON STATISTICS`	The AUTOTRACE report shows only the SQL statement execution statistics.
`SET AUTOTRACE ON`	The AUTOTRACE report includes both the optimizer execution path and the SQL statement execution statistics.
`SET AUTOTRACE TRACEONLY`	Like SET AUTOTRACE ON, but suppresses the printing of the user's query output, if any.

To use this feature, you must create a PLAN_TABLE table in your schema and then have the PLUSTRACE role granted to you. DBA privileges are required to grant the PLUSTRACE role. For information on how to grant a role and how to create the PLAN_TABLE table, see the *Oracle9i SQL Reference*. For more information about the roles and the PLAN_TABLE, see the *Oracle9i SQL Reference* and the AUTOTRACE variable of the SET command in Chapter 8.

Example 3–20 Creating a PLAN_TABLE

Run the following commands from your SQL*Plus session to create the PLAN_TABLE in the HR schema:

```
CONNECT HR/HR
@$ORACLE_HOME/RDBMS/ADMIN/UTLXPLAN.SQL
```

```
Table created.
```

Example 3–21 Creating the PLUSTRACE Role

Run the following commands from your SQL*Plus session to create the PLUSTRACE role and grant it to the DBA:

```
CONNECT / AS SYSDBA
@$ORACLE_HOME/SQLPLUS/ADMIN/PLUSTRCE.SQL
```

```
drop role plustrace;
Role dropped.
create role plustrace;
Role created.
.
.
.
grant plustrace to dba with admin option;
Grant succeeded.
```

Example 3–22 Granting the PLUSTRACE Role

Run the following commands from your SQL*Plus session to grant the PLUSTRACE role to the HR user:

```
CONNECT / AS SYSDBA
GRANT PLUSTRACE TO HR;
Grant succeeded.
```

Execution Plan

The Execution Plan shows the SQL optimizer's query execution path. Both tables are accessed by a full table scan, sorted, and then merged.

Each line of the Execution Plan has a sequential line number. SQL*Plus also displays the line number of the parent operation.

The Execution Plan consists of four columns displayed in the following order:

Column Name	Description
ID_PLUS_EXP	Shows the line number of each execution step.
PARENT_ID_PLUS_EXP	Shows the relationship between each step and its parent. This column is useful for large reports.
PLAN_PLUS_EXP	Shows each step of the report.
OBJECT_NODE_PLUS_EXP	Shows database links or parallel query servers used.

The format of the columns may be altered with the COLUMN command. For example, to stop the PARENT_ID_PLUS_EXP column being displayed, enter

```
COLUMN PARENT_ID_PLUS_EXP NOPRINT
```

The default formats can be found in the site profile (for example, glogin.sql).

The Execution Plan output is generated using the EXPLAIN PLAN command. For information about interpreting the output of EXPLAIN PLAN, see the *Oracle9i Performance Guide and Reference*.

Statistics

The statistics are recorded by the server when your statement executes and indicate the system resources required to execute your statement.

The *client* referred to in the statistics is SQL*Plus. *Oracle Net* refers to the generic process communication between SQL*Plus and the server, regardless of whether Oracle Net is installed.

You cannot change the default format of the statistics report.

For more information about the statistics and how to interpret them, see the *Oracle9i Performance Guide and Reference*.

Example 3–23 Tracing Statements for Performance Statistics and Query Execution Path

If the SQL buffer contains the following statement:

```
SELECT E.LAST_NAME, E.SALARY, J.JOB_TITLE
FROM EMPLOYEES E, JOBS J
WHERE E.JOB_ID=J.JOB_ID AND E.SALARY>12000
```

The statement can be automatically traced when it is run:

```
SET AUTOTRACE ON
/
```

```
LAST_NAME                 SALARY JOB_TITLE
------------------------- ---------- ----------------------------------
King                       24000 President
Kochhar                    17000 Administration Vice President
De Haan                    17000 Administration Vice President
Russell                    14000 Sales Manager
```

```
Partners                       13500 Sales Manager
Hartstein                      13000 Marketing Manager

6 rows selected.


Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (BY INDEX ROWID) OF 'EMPLOYEES'
   2    1     NESTED LOOPS
   3    2       TABLE ACCESS (FULL) OF 'JOBS'
   4    2       INDEX (RANGE SCAN) OF 'EMP_JOB_IX' (NON-UNIQUE)



Statistics
----------------------------------------------------------
          0  recursive calls
          2  db block gets
         34  consistent gets
          0  physical reads
          0  redo size
        848  bytes sent via SQL*Net to client
        503  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed
```

Note: Your output may vary depending on the version of the server to which you are connected and the configuration of the server.

Example 3–24 Tracing Statements Without Displaying Query Data

To trace the same statement without displaying the query data, enter:

```
SET AUTOTRACE TRACEONLY
/
```

```
6 rows selected.


Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE
   1    0   TABLE ACCESS (BY INDEX ROWID) OF 'EMPLOYEES'
   2    1     NESTED LOOPS
   3    2       TABLE ACCESS (FULL) OF 'JOBS'
   4    2       INDEX (RANGE SCAN) OF 'EMP_JOB_IX' (NON-UNIQUE)



Statistics
----------------------------------------------------------
          0  recursive calls
          2  db block gets
         34  consistent gets
          0  physical reads
          0  redo size
        848  bytes sent via SQL*Net to client
        503  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed
```

This option is useful when you are tuning a large query, but do not want to see the query report.

Example 3–25 Tracing Statements Using a Database Link

To trace a statement using a database link, enter:

```
SET AUTOTRACE TRACEONLY EXPLAIN
SELECT * FROM EMPLOYEES@MY_LINK;
```

```
Execution Plan
----------------------------------------------------------
0      SELECT STATEMENT (REMOTE) Optimizer=CHOOSE
1    0   TABLE ACCESS (FULL) OF 'EMPLOYEES'  MY_LINK.DB_DOMAIN
```

The Execution Plan shows that the table being accessed on line 1 is via the database link MY_LINK.DB_DOMAIN.

Tracing Parallel and Distributed Queries

When you trace a statement in a parallel or distributed query, the Execution Plan shows the cost based optimizer estimates of the number of rows (the *cardinality*). In general, the cost, cardinality and bytes at each node represent cumulative results. For example, the cost of a join node accounts for not only the cost of completing the join operations, but also the entire costs of accessing the relations in that join.

Lines marked with an asterisk (*) denote a parallel or remote operation. Each operation is explained in the second part of the report. See the *Oracle9i Performance Guide and Reference* for more information on parallel and distributed operations.

The second section of this report consists of three columns displayed in the following order

Column Name	Description
ID_PLUS_EXP	Shows the line number of each execution step.
OTHER_TAG_PLUS_EXP	Describes the function of the SQL statement in the OTHER_PLUS_EXP column.
OTHER_PLUS_EXP	Shows the text of the query for the parallel server or remote database.

The format of the columns may be altered with the COLUMN command. The default formats can be found in the site profile (for example, glogin.sql).

> **Note:** You must have Oracle7, Release 7.3 or greater to view the second section of this report.

Example 3–26 Tracing Statements With Parallel Query Option

To trace a parallel query running the parallel query option:

```
CREATE TABLE D2_T1 (UNIQUE1 NUMBER) PARALLEL -
(DEGREE 6);
```

```
Table created.
```

```
CREATE TABLE D2_T2 (UNIQUE1 NUMBER) PARALLEL -
(degree 6);
```

```
Table created.
```

```
CREATE UNIQUE INDEX D2_I_UNIQUE1 ON D2_T1(UNIQUE1);
```

```
Index created.
```

```
SET LONG 500 LONGCHUNKSIZE 500
SET AUTOTRACE ON EXPLAIN
SELECT /*+ INDEX(B,D2_I_UNIQUE1) USE_NL(B) ORDERED -
*/ COUNT (A.UNIQUE1)
FROM D2_T2 A, D2_T1 B
WHERE A.UNIQUE1 = B.UNIQUE1;
```

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=1 Card=1 Bytes=26)
   1    0   SORT (AGGREGATE)
   2    1     SORT* (AGGREGATE)                                        :Q2000
   3    2       NESTED LOOPS* (Cost=1 Card=41 Bytes=1066)              :Q2000
   4    3         TABLE ACCESS* (FULL) OF 'D2_T2' (Cost=1 Card=41 Byte :Q2000
          s=533)

   5    3         INDEX* (UNIQUE SCAN) OF 'D2_I_UNIQUE1' (UNIQUE)      :Q2000

   2 PARALLEL_TO_SERIAL            SELECT /*+ PIV_SSF */ SYS_OP_MSR(COUNT(A1.C0
                                   )) FROM (SELECT /*+ ORDERED NO_EXPAND USE_NL
                                   (A3) INDEX(A3 "D2_I_UNIQUE1") */ A2.C0 C0,A3
                                   .ROWID C1,A3."UNIQUE1" C2 FROM (SELECT /*+ N
                                   O_EXPAND ROWID(A4) */ A4."UNIQUE1" C0 FROM "
                                   D2_T2" PX_GRANULE(0, BLOCK_RANGE, DYNAMIC)
                                   A4) A2,"D2_T1" A3 WHERE A2.C0=A3."UNIQUE1")
                                   A1

   3 PARALLEL_COMBINED_WITH_PARENT
   4 PARALLEL_COMBINED_WITH_PARENT
   5 PARALLEL_COMBINED_WITH_PARENT
```

Line 0 of the Execution Plan shows the cost based optimizer estimates the number of rows at 1, taking 26 bytes. The total cost of the statement is 1.

Lines 2, 3, 4 and 5 are marked with asterisks, denoting parallel operations. For example, the NESTED LOOPS step on line 3 is a PARALLEL_TO_SERIAL operation. PARALLEL_TO_SERIAL operations execute a SQL statement to produce output serially. Line 2 also shows that the parallel query server had the identifier *Q2000*.

4

Formatting Query Results

This chapter explains how to format your query results to produce a finished report. This chapter covers the following topics:

- Formatting Columns
- Clarifying Your Report with Spacing and Summary Lines
- Defining Page and Report Titles and Dimensions
- Storing and Printing Query Results
- Creating Web Reports

Read this chapter while sitting at your computer and try out the examples shown. Before beginning, make sure you have access to the HR sample schema described in Chapter 1, "Introduction".

Formatting Columns

Through the SQL*Plus COLUMN command, you can change the column headings and reformat the column data in your query results.

Changing Column Headings

When displaying column headings, you can either use the default heading or you can change it using the COLUMN command. The following sections describe how default headings are derived and how to alter them using the COLUMN command. See the COLUMN command in Chapter 8 for more details.

Default Headings

SQL*Plus uses column or expression names as default column headings when displaying query results. Column names are often short and cryptic, however, and expressions can be hard to understand.

Changing Default Headings

You can define a more useful column heading with the HEADING clause of the COLUMN command, in the format shown below:

```
COLUMN column_name HEADING column_heading
```

Example 4–1 Changing a Column Heading

To produce a report from EMP_DETAILS_VIEW with new headings specified for LAST_NAME, SALARY, and COMMISSION_PCT, enter the following commands:

```
COLUMN LAST_NAME        HEADING 'LAST NAME'
COLUMN SALARY           HEADING 'MONTHLY SALARY'
COLUMN COMMISSION_PCT   HEADING COMMISSION
SELECT LAST_NAME, SALARY, COMMISSION_PCT
FROM EMP_DETAILS_VIEW
WHERE JOB_ID='SA_MAN'
```

```
LAST_NAME                 MONTHLY SALARY COMMISSION
------------------------- -------------- ----------
Russell                            14000         .4
Partners                           13500         .3
Errazuriz                          12000         .3
Cambrault                          11000         .3
Zlotkey                            10500         .2
```

Note: The new headings will remain in effect until you enter different headings, reset each column's format, or exit from SQL*Plus.

To change a column heading to two or more words, enclose the new heading in single or double quotation marks when you enter the COLUMN command. To display a column heading on more than one line, use a vertical bar (|) where you want to begin a new line. (You can use a character other than a vertical bar by changing the setting of the HEADSEP variable of the SET command. See the SET command in Chapter 8 for more information.)

Example 4–2 Splitting a Column Heading

To give the columns SALARY and LAST_NAME the headings MONTHLY SALARY and LAST NAME respectively, and to split the new headings onto two lines, enter

```
COLUMN SALARY HEADING 'MONTHLY|SALARY'
COLUMN LAST_NAME HEADING 'LAST|NAME'
```

Now rerun the query with the slash (/) command:

```
/
```

```
LAST                          MONTHLY
NAME                           SALARY COMMISSION
------------------------- ---------- ----------
Russell                         14000         .4
Partners                        13500         .3
Errazuriz                       12000         .3
Cambrault                       11000         .3
Zlotkey                         10500         .2
```

To change the character used to underline each column heading, set the UNDERLINE variable of the SET command to the desired character.

Example 4–3 Setting the Underline Character

To change the character used to underline headings to an equal sign and rerun the query, enter the following commands:

```
SET UNDERLINE =
/
```

```
LAST                          MONTHLY
NAME                           SALARY COMMISSION
========================= ========== ==========
Russell                         14000         .4
Partners                        13500         .3
Errazuriz                       12000         .3
Cambrault                       11000         .3
Zlotkey                         10500         .2
```

Now change the underline character back to a dash:

```
SET UNDERLINE '-'
```

Note: You must enclose the dash in quotation marks; otherwise, SQL*Plus interprets the dash as a hyphen indicating that you wish to continue the command on another line.

Formatting NUMBER Columns

When displaying NUMBER columns, you can either accept the SQL*Plus default display width or you can change it using the COLUMN command. The sections below describe the default display and how you can alter the default with the COLUMN command.

Default Display

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 normally displays numbers with as many digits as are required for accuracy, up to a standard display width determined by the value of the NUMWIDTH variable of the SET command (normally 10). If a number is larger than the value of SET NUMWIDTH, SQL*Plus rounds the number up or down to the maximum number of characters allowed.

You can choose a different format for any NUMBER column by using a format model in a COLUMN command. A format model is a representation of the way you want the numbers in the column to appear, using 9s to represent digits.

Changing the Default Display

The COLUMN command identifies the column you want to format and the model you want to use, as shown below:

```
COLUMN column_name FORMAT model
```

Use format models to add commas, dollar signs, angle brackets (around negative values), and / or leading zeros to numbers in a given column. You can also round the values to a given number of decimal places, display minus signs to the right of negative values (instead of to the left), and display values in exponential notation.

To use more than one format model for a single column, combine the desired models in one COLUMN command (see Example 4–4). For a complete list of format models and further details, see the COLUMN command in Chapter 8.

Example 4–4 Formatting a NUMBER Column

To display SALARY with a dollar sign, a comma, and the numeral zero instead of a blank for any zero values, enter the following command:

```
COLUMN SALARY FORMAT $99,990
```

Now rerun the current query:

```
/
```

```
LAST                      MONTHLY
NAME                       SALARY COMMISSION
------------------------- -------- ----------
Russell                   $14,000         .4
Partners                  $13,500         .3
Errazuriz                 $12,000         .3
Cambrault                 $11,000         .3
Zlotkey                   $10,500         .2
```

Use a zero in your format model, as shown above, when you use other formats such as a dollar sign and wish to display a zero in place of a blank for zero values.

Note: The format model will stay in effect until you enter a new one, reset the column's format with

```
COLUMN column_name CLEAR
```

or exit from SQL*Plus.

Formatting Datatypes

When displaying datatypes, you can either accept the SQL*Plus default display width or you can change it using the COLUMN command. Datatypes, in this manual, include the following types:

- CHAR
- NCHAR
- VARCHAR2 (VARCHAR)
- NVARCHAR2 (NCHAR VARYING)
- DATE
- LONG
- CLOB
- NCLOB

Default Display

The default width of datatype columns is the width of the column in the database.

The default width and format of unformatted DATE columns in SQL*Plus is derived from the NLS parameters in effect. Otherwise, the default format width is A9. For more information on formatting DATE columns, see the FORMAT clause of the COLUMN command in Chapter 8.

Left justification is the default for datatypes.

Changing the Default Display

You can change the displayed width of a datatype or DATE, by using the COLUMN command with a format model consisting of the letter A (for alphanumeric) followed by a number representing the width of the column in characters.

Within the COLUMN command, identify the column you want to format and the model you want to use:

```
COLUMN column_name FORMAT model
```

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. See the COLUMN command in Chapter 8 for more details.

Example 4–5 Formatting a Character Column

To set the width of the column LAST_NAME to four characters and rerun the current query, enter

```
COLUMN LAST_NAME FORMAT A4
/
```

```
LAST  MONTHLY
NAME   SALARY COMMISSION
---- -------- ----------
Russ  $14,000         .4
ell

Part  $13,500         .3
ners

Erra  $12,000         .3
zuri
z


LAST  MONTHLY
NAME   SALARY COMMISSION
---- -------- ----------
Camb  $11,000         .3
raul
t

Zlot  $10,500         .2
key
```

Note: The format model will stay in effect until you enter a new one, reset the column's format with

`COLUMN` *`column_name`* `CLEAR`

or exit from SQL*Plus.

If the WRAP variable of the SET command is set to ON (its default value), the employee names wrap to the next line after the fourth character, as shown in Example 4–5. If WRAP is set to OFF, the names are truncated (cut off) after the fourth character.

The system variable WRAP controls all columns; you can override the setting of WRAP for a given column through the WRAPPED, WORD_WRAPPED, and TRUNCATED clauses of the COLUMN command. See the COLUMN command in Chapter 8 for more information on these clauses. You will use the WORD_WRAPPED clause of COLUMN later in this chapter.

> **Note:** The column heading is truncated regardless of the setting of WRAP or any COLUMN command clauses.

Now return the column to its previous format:

```
COLUMN LAST_NAME FORMAT A10
```

Copying Column Display Attributes

When you want to give more than one column the same display attributes, you can reduce the length of the commands you must enter by using the LIKE clause of the COLUMN command. The LIKE clause tells SQL*Plus to copy the display attributes of a previously defined column to the new column, except for changes made by other clauses in the same command.

Example 4–6 Copying a Column's Display Attributes

To give the column COMMISSION_PCT the same display attributes you gave to SALARY, but to specify a different heading, enter the following command:

```
COLUMN COMMISSION_PCT LIKE SALARY HEADING BONUS
```

Rerun the query:

```
/
```

```
LAST        MONTHLY
NAME         SALARY    BONUS
---------- -------- --------
Russell     $14,000       $0
Partners    $13,500       $0
Errazuriz   $12,000       $0
Cambrault   $11,000       $0
Zlotkey     $10,500       $0
```

Listing and Resetting Column Display Attributes

To list the current display attributes for a given column, use the COLUMN command followed by the column name only, as shown below:

```
COLUMN column_name
```

To list the current display attributes for all columns, enter the COLUMN command with no column names or clauses after it:

```
COLUMN
```

To reset the display attributes for a column to their default values, use the CLEAR clause of the COLUMN command as shown below:

```
COLUMN column_name CLEAR
```

To reset the attributes for all columns, use the COLUMNS clause of the CLEAR command.

Example 4–7 Resetting Column Display Attributes to their Defaults

To reset all columns' display attributes to their default values, enter the following command:

```
CLEAR COLUMNS
```

```
columns cleared
```

Suppressing and Restoring Column Display Attributes

You can suppress and restore the display attributes you have given a specific column. To suppress a column's display attributes, enter a COLUMN command in the following form:

```
COLUMN column_name OFF
```

The OFF clause tells SQL*Plus to use the default display attributes for the column, but does not remove the attributes you have defined through the COLUMN command. To restore the attributes you defined through COLUMN, use the ON clause:

```
COLUMN column_name ON
```

Printing a Line of Characters after Wrapped Column Values

As you have seen, by default SQL*Plus wraps column values to additional lines when the value does not fit the column width. If you want to insert a *record separator* (a line of characters or a blank line) after each wrapped line of output (or after every row), use the RECSEP and RECSEPCHAR variables of the SET command.

RECSEP determines when the line of characters is printed; you set RECSEP to EACH to print after every line, to WRAPPED to print after wrapped lines, and to OFF to suppress printing. The default setting of RECSEP is WRAPPED.

RECSEPCHAR sets the character printed in each line. You can set RECSEPCHAR to any character.

You may wish to wrap whole words to additional lines when a column value wraps to additional lines. To do so, use the WORD_WRAPPED clause of the COLUMN command as shown below:

```
COLUMN column_name WORD_WRAPPED
```

Example 4–8 Printing a Line of Characters after Wrapped Column Values

To print a line of dashes after each wrapped column value, enter the commands:

```
SET RECSEP WRAPPED
SET RECSEPCHAR "-"
```

Finally, enter the following query:

```
SELECT LAST_NAME, JOB_TITLE, CITY
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000;
```

Now restrict the width of the column JOB_TITLE and tell SQL*Plus to wrap whole words to additional lines when necessary:

```
COLUMN JOB_TITLE FORMAT A20 WORD_WRAPPED
```

Run the query:

```
/
```

```
LAST_NAME                 JOB_TITLE            CITY
------------------------- -------------------- ------------------------------
King                      President            Seattle
Kochhar                   Administration Vice  Seattle
                          President
-------------------------------------------------------------------------------
```

```
De Haan                   Administration Vice  Seattle
                          President
-------------------------------------------------------------------------------
Russell                   Sales Manager        Oxford
Partners                  Sales Manager        Oxford
Hartstein                 Marketing Manager    Toronto

6 rows selected.
```

If you set RECSEP to EACH, SQL*Plus prints a line of characters after every row (after every department, for the above example).

Before continuing, set RECSEP to OFF to suppress the printing of record separators:

```
SET RECSEP OFF
```

Clarifying Your Report with Spacing and Summary Lines

When you use an ORDER BY clause in your SQL SELECT command, rows with the same value in the ordered column (or expression) are displayed together in your output. You can make this output more useful to the user by using the SQL*Plus BREAK and COMPUTE commands to create subsets of records and add space and / or summary lines after each subset.

The column you specify in a BREAK command is called a *break column*. By including the break column in your ORDER BY clause, you create meaningful subsets of records in your output. You can then add formatting to the subsets within the same BREAK command, and add a summary line (containing totals, averages, and so on) by specifying the break column in a COMPUTE command.

For example, the following query, without BREAK or COMPUTE commands,

```
SELECT DEPARTMENT_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY > 12000
ORDER BY DEPARTMENT_ID;
```

```
DEPARTMENT_ID LAST_NAME                    SALARY
------------- ------------------------- ----------
           20 Hartstein                      13000
           80 Russell                        14000
           80 Partners                       13500
           90 King                           24000
```

```
           90 Kochhar                       17000
           90 De Haan                       17000

6 rows selected.
```

To make this report more useful, you would use BREAK to establish DEPARTMENT_ID as the break column. Through BREAK you could suppress duplicate values in DEPARTMENT_ID and place blank lines or begin a new page between departments. You could use BREAK in conjunction with COMPUTE to calculate and print summary lines containing the total (and / or average, maximum, minimum, standard deviation, variance, or count of rows of) salary for each department and for all departments.

Suppressing Duplicate Values in Break Columns

The BREAK command suppresses duplicate values by default in the column or expression you name. Thus, to suppress the duplicate values in a column specified in an ORDER BY clause, use the BREAK command in its simplest form:

```
BREAK ON break_column
```

Note: Whenever you specify a column or expression in a BREAK command, use an ORDER BY clause specifying the same column or expression. If you do not do this, breaks occur every time the column value changes.

Example 4–9 Suppressing Duplicate Values in a Break Column

To suppress the display of duplicate department numbers in the query results shown above, enter the following commands:

```
BREAK ON DEPARTMENT_ID;
```

For the following query (which is the current query stored in the buffer):

```
SELECT DEPARTMENT_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY > 12000
ORDER BY DEPARTMENT_ID;
```

```
DEPARTMENT_ID LAST_NAME                SALARY
------------- ------------------------ ----------
           20 Hartstein                       13000
```

```
           80 Russell                        14000
              Partners                       13500
           90 King                           24000
              Kochhar                        17000
              De Haan                        17000

6 rows selected.
```

Inserting Space when a Break Column's Value Changes

You can insert blank lines or begin a new page each time the value changes in the break column. To insert *n* blank lines, use the BREAK command in the following form:

```
BREAK ON break_column SKIP n
```

To skip a page, use the command in this form:

```
BREAK ON break_column SKIP PAGE
```

Example 4–10 Inserting Space when a Break Column's Value Changes

To place one blank line between departments, enter the following command:

```
BREAK ON DEPARTMENT_ID SKIP 1
```

Now rerun the query:

```
/
```

```
DEPARTMENT_ID LAST_NAME                    SALARY
------------- ------------------------ ----------
           20 Hartstein                      13000

           80 Russell                        14000
              Partners                       13500

           90 King                           24000
              Kochhar                        17000
              De Haan                        17000

6 rows selected.
```

Inserting Space after Every Row

You may wish to insert blank lines or a blank page after every row. To skip *n* lines after every row, use BREAK in the following form:

```
BREAK ON ROW SKIP n
```

To skip a page after every row, use

```
BREAK ON ROW SKIP PAGE
```

> **Note:** SKIP PAGE does not cause a physical page break character to be generated unless you have also specified NEWPAGE 0.

Using Multiple Spacing Techniques

Suppose you have more than one column in your ORDER BY clause and wish to insert space when each column's value changes. Each BREAK command you enter replaces the previous one. Thus, if you want to use different spacing techniques in one report or insert space after the value changes in more than one ordered column, you must specify multiple columns and actions in a single BREAK command.

Example 4–11 Combining Spacing Techniques

First, clear the buffer:

```
CLEAR BUFFER
```

Type the following:

```
SELECT DEPARTMENT_ID, JOB_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000
ORDER BY DEPARTMENT_ID, JOB_ID;
```

Now, to skip a page when the value of DEPARTMENT_ID changes and one line when the value of JOB_ID changes, enter the following command:

```
BREAK ON DEPARTMENT_ID SKIP PAGE ON JOB_ID SKIP 1
```

To show that SKIP PAGE has taken effect, create a TTITLE with a page number:

```
TTITLE COL 35 FORMAT 9 'Page:' SQL.PNO
```

Run the new query to see the results:

```
                                    Page: 1
DEPARTMENT_ID JOB_ID     LAST_NAME                     SALARY
------------- ---------- ------------------------- ----------
           20 MK_MAN     Hartstein                      13000

                                    Page: 2
DEPARTMENT_ID JOB_ID     LAST_NAME                     SALARY
------------- ---------- ------------------------- ----------
           80 SA_MAN     Russell                        14000
                         Partners                       13500

                                    Page: 3
DEPARTMENT_ID JOB_ID     LAST_NAME                     SALARY
------------- ---------- ------------------------- ----------
           90 AD_PRES    King                           24000

              AD_VP      Kochhar                        17000
                         De Haan                        17000

6 rows selected.
```

Listing and Removing Break Definitions

Before continuing, turn off the top title display without changing its definition:

```
TTITLE OFF
```

You can list your current break definition by entering the BREAK command with no clauses:

```
BREAK
```

You can remove the current break definition by entering the CLEAR command with the BREAKS clause:

```
CLEAR BREAKS
```

You may wish to place the command CLEAR BREAKS at the beginning of every command file to ensure that previously entered BREAK commands will not affect queries you run in a given file.

Computing Summary Lines when a Break Column's Value Changes

If you organize the rows of a report into subsets with the BREAK command, you can perform various computations on the rows in each subset. You do this with the functions of the SQL*Plus COMPUTE command. Use the BREAK and COMPUTE commands together in the following forms:

```
BREAK ON break_column
COMPUTE function LABEL label_name OF column column column
... ON break_column
```

You can include multiple break columns and actions, such as skipping lines in the BREAK command, as long as the column you name after ON in the COMPUTE command also appears after ON in the BREAK command. To include multiple break columns and actions in BREAK when using it in conjunction with COMPUTE, use these commands in the following forms:

```
BREAK ON break_column_1 SKIP PAGE ON break_column_2 SKIP 1
COMPUTE function LABEL label_name OF column column column
... ON break_column_2
```

The COMPUTE command has no effect without a corresponding BREAK command.

You can COMPUTE on NUMBER columns and, in certain cases, on all types of columns. For more information about the COMPUTE command, see the "Command Reference" in Chapter 8.

The following table lists compute functions and their effects

Table 4–1 Compute Functions

Function	Effect
SUM	Computes the sum of the values in the column.
MINIMUM	Computes the minimum value in the column.
MAXIMUM	Computes the maximum value in the column.
AVG	Computes the average of the values in the column.
STD	Computes the standard deviation of the values in the column.
VARIANCE	Computes the variance of the values in the column.
COUNT	Computes the number of non-null values in the column.
NUMBER	Computes the number of rows in the column.

The function you specify in the COMPUTE command applies to all columns you enter after OF and before ON. The computed values print on a separate line when the value of the ordered column changes.

Labels for ON REPORT and ON ROW computations appear in the first column; otherwise, they appear in the column specified in the ON clause.

You can change the compute label by using COMPUTE LABEL. If you do not define a label for the computed value, SQL*Plus prints the unabbreviated function keyword.

The compute label can be suppressed by using the NOPRINT option of the COLUMN command on the break column. See the COMPUTE command in Chapter 8 for more details.

Example 4–12 Computing and Printing Subtotals

To compute the total of SALARY by department, first list the current BREAK definition:

```
BREAK
```

which displays current BREAK definitions:

```
break on DEPARTMENT_ID page  nodup
          on JOB_ID skip 1 nodup
```

Now enter the following COMPUTE command and run the current query:

```
COMPUTE SUM OF SALARY ON DEPARTMENT_ID
 /
```

```
DEPARTMENT_ID JOB_ID     LAST_NAME                    SALARY
------------- ---------- ------------------------ ----------
           20 MK_MAN     Hartstein                     13000

************* **********                           ----------
sum                                                    13000

DEPARTMENT_ID JOB_ID     LAST_NAME                    SALARY
------------- ---------- ------------------------ ----------
           80 SA_MAN     Russell                       14000
                         Partners                      13500

************* **********                           ----------
sum                                                    27500
```

```
DEPARTMENT_ID JOB_ID     LAST_NAME                     SALARY
------------- ---------- ------------------------- ----------
           90 AD_PRES    King                          24000

              AD_VP      Kochhar                       17000
                         De Haan                       17000

************* **********                           ----------
sum                                                    58000

6 rows selected.
```

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

```
COLUMN DUMMY NOPRINT;
COMPUTE SUM OF SALARY ON DUMMY;
BREAK ON DUMMY SKIP 1;
SELECT DEPARTMENT_ID DUMMY,DEPARTMENT_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000
ORDER BY DEPARTMENT_ID;
```

```
DEPARTMENT_ID LAST_NAME                     SALARY
------------- ------------------------- ----------
           20 Hartstein                      13000
                                        ----------
                                             13000

           80 Russell                        14000
           80 Partners                       13500
                                        ----------
                                             27500

           90 King                           24000
           90 Kochhar                        17000
           90 De Haan                        17000
                                        ----------
                                             58000


6 rows selected.
```

To compute the salaries just at the end of the report:

```
COLUMN DUMMY NOPRINT;
COMPUTE SUM OF SALARY ON DUMMY;
BREAK ON DUMMY;
SELECT NULL DUMMY,DEPARTMENT_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000
ORDER BY DEPARTMENT_ID;
```

```
DEPARTMENT_ID LAST_NAME                    SALARY
------------- ------------------------- ----------
           20 Hartstein                      13000
           80 Russell                        14000
           80 Partners                       13500
           90 King                           24000
           90 Kochhar                        17000
           90 De Haan                        17000
                                          ----------
                                             98500

6 rows selected.
```

Note: The format of the column SALARY controls the appearance of the sum of SALARY, as well as the individual values of SALARY. When you establish the format of a NUMBER column, you must allow for the size of sums you will include in your report.

Computing Summary Lines at the End of the Report

You can calculate and print summary lines based on all values in a column by using BREAK and COMPUTE in the following forms:

```
BREAK ON REPORT
COMPUTE function LABEL label_name OF column column column
... ON REPORT
```

Example 4–13 Computing and Printing a Grand Total

To calculate and print the grand total of salaries for all sales people and change the compute label, first enter the following BREAK and COMPUTE commands:

```
BREAK ON REPORT
COMPUTE SUM LABEL TOTAL OF SALARY ON REPORT
```

Next, enter and run a new query:

```
SELECT LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE JOB_ID='SA_MAN';
```

```
LAST_NAME                      SALARY
------------------------- ----------
Russell                         14000
Partners                        13500
Errazuriz                       12000
Cambrault                       11000
Zlotkey                         10500
                           ----------
TOTAL                           61000
```

To print a grand total (or grand average, grand maximum, and so on) in addition to subtotals (or sub-averages, and so on), include a break column and an ON REPORT clause in your BREAK command. Then, enter one COMPUTE command for the break column and another to compute ON REPORT:

```
BREAK ON break_column ON REPORT
COMPUTE function LABEL label_name OF column ON break_column
COMPUTE function LABEL label_name OF column ON REPORT
```

Computing Multiple Summary Values and Lines

You can compute and print the same type of summary value on different columns. To do so, enter a separate COMPUTE command for each column.

Example 4–14 Computing the Same Type of Summary Value on Different Columns

To print the total of salaries and commissions for all sales people, first enter the following COMPUTE command:

```
COMPUTE SUM OF SALARY COMMISSION_PCT ON REPORT
```

You do not have to enter a BREAK command; the BREAK you entered in Example 4–13 is still in effect. Now, change the first line of the select query to include COMMISSION_PCT:

```
1
  1* SELECT LAST_NAME, SALARY
APPEND , COMMISSION_PCT;
```

Finally, run the revised query to see the results:

```
/
```

```
LAST_NAME                     SALARY COMMISSION_PCT
------------------------- ---------- --------------
Russell                        14000             .4
Partners                       13500             .3
Errazuriz                      12000             .3
Cambrault                      11000             .3
Zlotkey                        10500             .2
                           ---------- --------------
sum                            61000            1.5
```

You can also print multiple summary lines on the same break column. To do so, include the function for each summary line in the COMPUTE command as follows:

```
COMPUTE function LABEL label_name function
  LABEL label_name function LABEL label_name ...
  OF column ON break_column
```

If you include multiple columns after OF and before ON, COMPUTE calculates and prints values for each column you specify.

Example 4–15 Computing Multiple Summary Lines on the Same Break Column

To compute the average and sum of salaries for the sales department, first enter the following BREAK and COMPUTE commands:

```
BREAK ON DEPARTMENT_ID
COMPUTE AVG SUM OF SALARY ON DEPARTMENT_ID
```

Now, enter and run the following query:

```
SELECT DEPARTMENT_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE DEPARTMENT_ID = 30
ORDER BY DEPARTMENT_ID, SALARY;
```

```
DEPARTMENT_ID LAST_NAME                     SALARY
------------- ------------------------- ----------
           30 Colmenares                      2500
              Himuro                          2600
              Tobias                          2800
              Baida                           2900
              Khoo                            3100
```

```
              Raphaely                        11000
*************                                ----------
avg                                            4150
sum                                           24900

6 rows selected.
```

Listing and Removing COMPUTE Definitions

You can list your current COMPUTE definitions by entering the COMPUTE command with no clauses:

```
COMPUTE
```

Example 4–16 Removing COMPUTE Definitions

To remove all COMPUTE definitions and the accompanying BREAK definition, enter the following commands:

```
CLEAR BREAKS
breaks cleared
CLEAR COMPUTES
computes cleared
```

You may wish to place the commands CLEAR BREAKS and CLEAR COMPUTES at the beginning of every command file to ensure that previously entered BREAK and COMPUTE commands will not affect queries you run in a given file.

Defining Page and Report Titles and Dimensions

The word *page* refers to a screen full of information on your display or a page of a spooled (printed) report. You can place top and bottom titles on each page, set the number of lines per page, and determine the width of each line.

The word *report* refers to the complete results of a query. You can also place headers and footers on each report and format them in the same way as top and bottom titles on pages.

Setting the Top and Bottom Titles and Headers and Footers

As you have already seen, you can set a title to display at the top of each page of a report. You can also set a title to display at the bottom of each page. The TTITLE command defines the top title; the BTITLE command defines the bottom title.

You can also set a header and footer for each report. The REPHEADER command defines the report header; the REPFOOTER command defines the report footer.

A TTITLE, BTITLE, REPHEADER or REPFOOTER command consists of the command name followed by one or more clauses specifying a position or format and a CHAR value you wish to place in that position or give that format. You can include multiple sets of clauses and CHAR values:

```
TTITLE position_clause(s) char_value position_clause(s) char_value ...
BTITLE position_clause(s) char_value position_clause(s) char_value ...
REPHEADER position_clause(s) char_value position_clause(s) char_value ...
REPFOOTER position_clause(s) char_value position_clause(s) char_value ...
```

For descriptions of all TTITLE, BTITLE, REPHEADER and REPFOOTER clauses, see the TTITLE and REPHEADER commands in Chapter 8.

Example 4–17 Placing a Top and Bottom Title on a Page

To put titles at the top and bottom of each page of a report, enter

```
TTITLE CENTER -
"ACME SALES DEPARTMENT PERSONNEL REPORT"
BTITLE CENTER "COMPANY CONFIDENTIAL"
```

Now run the current query:

```
/
```

```
                    ACME SALES DEPARTMENT PERSONNEL REPORT
DEPARTMENT_ID LAST_NAME                      SALARY
------------- ------------------------- ----------
           30 Colmenares                       2500
           30 Himuro                           2600
           30 Tobias                           2800
           30 Baida                            2900
           30 Khoo                             3100
           30 Raphaely                        11000


                           COMPANY CONFIDENTIAL

6 rows selected.
```

Example 4–18 Placing a Header on a Report

To put a report header on a separate page, and to center it, enter

```
REPHEADER PAGE CENTER 'PERFECT WIDGETS'
```

Now run the current query:

```
/
```

which displays the following two pages of output, with the new REPHEADER displayed on the first page:

```
                ACME SALES DEPARTMENT PERSONNEL REPORT
                           PERFECT WIDGETS


                                COMPANY CONFIDENTIAL

                ACME SALES DEPARTMENT PERSONNEL REPORT
DEPARTMENT_ID LAST_NAME                    SALARY
------------- ------------------------- ----------
           30 Colmenares                      2500
           30 Himuro                          2600
           30 Tobias                          2800
           30 Baida                           2900
           30 Khoo                            3100
           30 Raphaely                       11000

                                COMPANY CONFIDENTIAL

6 rows selected.
```

To suppress the report header without changing its definition, enter

```
REPHEADER OFF
```

Positioning Title Elements

The report in the preceding exercises might look more attractive if you give the company name more emphasis and place the type of report and the department name on either end of a separate line. It may also help to reduce the linesize and thus center the titles more closely around the data.

You can accomplish these changes by adding some clauses to the TTITLE command and by resetting the system variable LINESIZE, as the following example shows.

You can format report headers and footers in the same way as BTITLE and TTITLE using the REPHEADER and REPFOOTER commands.

Example 4–19 Positioning Title Elements

To redisplay the personnel report with a repositioned top title, enter the following commands:

```
TTITLE CENTER 'A C M E  W I D G E T' SKIP 1 -
CENTER ==================== SKIP 1 LEFT 'PERSONNEL REPORT' -
RIGHT 'SALES DEPARTMENT' SKIP 2
SET LINESIZE 60
/
```

```
                    A C M E  W I D G E T
                    ====================
PERSONNEL REPORT                  SALES DEPARTMENT

DEPARTMENT_ID LAST_NAME                     SALARY
------------- ------------------------- ----------
           30 Colmenares                      2500
           30 Himuro                          2600
           30 Tobias                          2800
           30 Baida                           2900
           30 Khoo                            3100
           30 Raphaely                       11000

                    COMPANY CONFIDENTIAL

6 rows selected.
```

The LEFT, RIGHT, and CENTER clauses place the following values at the beginning, end, and center of the line. The SKIP clause tells SQL*Plus to move down one or more lines.

Note that there is no longer any space between the last row of the results and the bottom title. The last line of the bottom title prints on the last line of the page. The amount of space between the last row of the report and the bottom title depends on the overall page size, the number of lines occupied by the top title, and the number of rows in a given page. In the above example, the top title occupies three more lines than the top title in the previous example. You will learn to set the number of lines per page later in this chapter.

To always print *n* blank lines before the bottom title, use the SKIP *n* clause at the beginning of the BTITLE command. For example, to skip one line before the bottom title in the example above, you could enter the following command:

```
BTITLE SKIP 1 CENTER 'COMPANY CONFIDENTIAL'
```

Indenting a Title Element

You can use the COL clause in TTITLE or BTITLE to indent the title element a specific number of spaces. For example, COL 1 places the following values in the first character position, and so is equivalent to LEFT, or an indent of zero. COL 15 places the title element in the 15th character position, indenting it 14 spaces.

Example 4–20 Indenting a Title Element

To print the company name left-aligned with the report name indented five spaces on the next line, enter

```
TTITLE LEFT 'ACME WIDGET' SKIP 1 -
COL 6 'SALES DEPARTMENT PERSONNEL REPORT' SKIP 2
```

Now rerun the current query to see the results:

```
/
```

```
ACME WIDGET
     SALES DEPARTMENT PERSONNEL REPORT

DEPARTMENT_ID LAST_NAME                    SALARY
------------- ------------------------ ----------
           30 Colmenares                     2500
           30 Himuro                         2600
           30 Tobias                         2800
           30 Baida                          2900
           30 Khoo                           3100
           30 Raphaely                      11000

                    COMPANY CONFIDENTIAL

6 rows selected.
```

Entering Long Titles

If you need to enter a title greater than 500 characters in length, you can use the SQL*Plus command DEFINE to place the text of each line of the title in a separate user variable:

```
DEFINE LINE1 = 'This is the first line...'
DEFINE LINE2 = 'This is the second line...'
DEFINE LINE3 = 'This is the third line...'
```

Then, reference the variables in your TTITLE or BTITLE command as follows:

```
TTITLE CENTER LINE1 SKIP 1 CENTER LINE2 SKIP 1 -
CENTER LINE3
```

Displaying the Page Number and other System-Maintained Values in Titles

You can display the current page number and other system-maintained values in your title by entering a system value name as a title element, for example:

```
TTITLE LEFT system-maintained_value_name
```

There are five system-maintained values you can display in titles, the most commonly used of which is SQL.PNO (the current page number). For a list of system-maintained values you can display in titles, see the TTITLE command in the "Command Reference" in Chapter 8.

Example 4–21 Displaying the Current Page Number in a Title

To display the current page number at the top of each page, along with the company name, enter the following command:

```
TTITLE LEFT 'ACME WIDGET' RIGHT 'PAGE:' SQL.PNO SKIP 2
```

Now rerun the current query:

```
/
```

```
ACMEWIDGET                                PAGE:         1

DEPARTMENT_ID LAST_NAME                   SALARY
------------- ------------------------ ----------
           30 Colmenares                     2500
           30 Himuro                         2600
           30 Tobias                         2800
           30 Baida                          2900
           30 Khoo                           3100
```

```
            30 Raphaely                    11000


                      COMPANY CONFIDENTIAL

6 rows selected.
```

Note that SQL.PNO has a format ten spaces wide. You can change this format with the FORMAT clause of TTITLE (or BTITLE).

Example 4–22 Formatting a System-Maintained Value in a Title

To close up the space between the word PAGE: and the page number, reenter the TTITLE command as shown:

```
TTITLE LEFT 'ACME WIDGET' RIGHT 'PAGE:' FORMAT 999 -
SQL.PNO SKIP 2
```

Now rerun the query:

```
/
```

```
ACME WIDGET                                    'PAGE:'   1

DEPARTMENT_ID LAST_NAME                    SALARY
------------- ------------------------ ----------
           30 Colmenares                     2500
           30 Himuro                         2600
           30 Tobias                         2800
           30 Baida                          2900
           30 Khoo                           3100
           30 Raphaely                      11000


                      COMPANY CONFIDENTIAL

6 rows selected.
```

Listing, Suppressing, and Restoring Page Title Definitions

To list a page title definition, enter the appropriate title command with no clauses:

```
TTITLE
BTITLE
```

To suppress a title definition, enter:

```
TTITLE OFF
BTITLE OFF
```

These commands cause SQL*Plus to cease displaying titles on reports, but do not clear the current definitions of the titles. You may restore the current definitions by entering:

```
TTITLE ON
BTITLE ON
```

Displaying Column Values in Titles

You may wish to create a master / detail report that displays a changing master column value at the top of each page with the detail query results for that value below. You can reference a column value in a top title by storing the desired value in a variable and referencing the variable in a TTITLE command. Use the following form of the COLUMN command to define the variable:

COLUMN *column_name* NEW_VALUE *variable_name*

You must include the master column in an ORDER BY clause and in a BREAK command using the SKIP PAGE clause.

Example 4–23 Creating a Master/Detail Report

Suppose you want to create a report that displays two different managers' employee numbers, each at the top of a separate page, and the people reporting to the manager on the same page as the manager's employee number. First create a variable, MGRVAR, to hold the value of the current manager's employee number:

```
COLUMN MANAGER_ID NEW_VALUE MGRVAR NOPRINT
```

Because you will only display the managers' employee numbers in the title, you do not want them to print as part of the detail. The NOPRINT clause you entered above tells SQL*Plus not to print the column MANAGER_ID.

Next, include a label and the value in your page title, enter the proper BREAK command, and suppress the bottom title from the last example:

```
TTITLE LEFT 'Manager: ' MGRVAR SKIP 2
BREAK ON MANAGER_ID SKIP PAGE
BTITLE OFF
```

Finally, enter and run the following query:

```
SELECT MANAGER_ID, DEPARTMENT_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE MANAGER_ID IN (101, 201)
ORDER BY MANAGER_ID, DEPARTMENT_ID;
```

```
Manager:       101

DEPARTMENT_ID LAST_NAME                    SALARY
------------- ------------------------ ----------
           10 Whalen                         4400
           40 Mavris                         6500
           70 Baer                          10000
          100 Greenberg                     12000
          110 Higgins                       12000

Manager:       201

DEPARTMENT_ID LAST_NAME                    SALARY
------------- ------------------------ ----------
           20 Fay                            6000

6 rows selected.
```

If you want to print the value of a column at the bottom of the page, you can use the COLUMN command in the following form:

```
COLUMN column_name OLD_VALUE variable_name
```

SQL*Plus prints the bottom title as part of the process of breaking to a new page—after finding the new value for the master column. Therefore, if you simply referenced the NEW_VALUE of the master column, you would get the value for the next set of details. OLD_VALUE remembers the value of the master column that was in effect before the page break began.

Displaying the Current Date in Titles

You can, of course, date your reports by simply typing a value in the title. This is satisfactory for ad hoc reports, but if you want to run the same report repeatedly, you would probably prefer to have the date automatically appear when the report is run. You can do this by creating a variable to hold the current date.

To create the variable (in this example named _DATE), you can add the following commands to your SQL*Plus LOGIN file:

```
SET TERMOUT OFF
BREAK ON TODAY
COLUMN TODAY NEW_VALUE _DATE
SELECT TO_CHAR(SYSDATE, 'fmMonth DD, YYYY') TODAY
FROM DUAL;
CLEAR BREAKS
SET TERMOUT ON
```

When you start SQL*Plus, these commands place the value of SYSDATE (the current date) into a variable named _DATE. To display the current date, you can reference _DATE in a title as you would any other variable.

The date format model you include in the SELECT command in your LOGIN file determines the format in which SQL*Plus displays the date. See your *Oracle9i SQL Reference* for more information on date format models. For more information about the LOGIN file, see the section "Modifying Your LOGIN File" in Chapter 3.

You can also enter these commands interactively at the command prompt. For more information, see the COLUMN command in Chapter 8.

Setting Page Dimensions

Typically, a page of a report contains the number of blank line(s) set in the NEWPAGE variable of the SET command, a top title, column headings, your query results, and a bottom title. SQL*Plus displays a report that is too long to fit on one page on several consecutive pages, each with its own titles and column headings. The amount of data SQL*Plus displays on each page depends on the current page dimensions.

The default page dimensions used by SQL*Plus are shown below:

- number of lines before the top title: 1
- number of lines per page, from the top title to the bottom of the page: 24
- number of characters per line: 80

You can change these settings to match the size of your computer screen or, for printing, the size of a sheet of paper.

You can change the page length with the system variable PAGESIZE. For example, you may wish to do so when you print a report, since printed pages are customarily 66 lines long.

To set the number of lines between the beginning of each page and the top title, use the NEWPAGE variable of the SET command:

```
SET NEWPAGE number_of_lines
```

If you set NEWPAGE to zero, SQL*Plus skips zero lines and displays and prints a formfeed character to begin a new page. On most types of computer screens, the formfeed character clears the screen and moves the cursor to the beginning of the first line. When you print a report, the formfeed character makes the printer move to the top of a new sheet of paper, even if the overall page length is less than that of the paper. If you set NEWPAGE to NONE, SQL*Plus does not print a blank line or formfeed between report pages.

To set the number of lines on a page, use the PAGESIZE variable of the SET command:

```
SET PAGESIZE number_of_lines
```

You may wish to reduce the linesize to center a title properly over your output, or you may want to increase linesize for printing on wide paper. You can change the line width using the LINESIZE variable of the SET command:

```
SET LINESIZE number_of_characters
```

Example 4–24 Setting Page Dimensions

To set the page size to 66 lines, clear the screen (or advance the printer to a new sheet of paper) at the start of each page, and set the linesize to 70, enter the following commands:

```
SET PAGESIZE 66
SET NEWPAGE 0
SET LINESIZE 70
```

Now enter and run the following commands to see the results:

```
TTITLE CENTER 'ACME WIDGET PERSONNEL REPORT' SKIP 1 -
CENTER '01-JAN-2001' SKIP 2
```

Now run the following query:

```
COLUMN FIRST_NAME HEADING 'FIRST|NAME';
COLUMN LAST_NAME HEADING 'LAST|NAME';
COLUMN SALARY  HEADING 'MONTHLY|SALARY' FORMAT $99,999;
SELECT DEPARTMENT_ID, FIRST_NAME, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000;
```

```
                    ACME WIDGET PERSONNEL REPORT
                              01-JAN-2001

              FIRST                LAST                     MONTHLY
DEPARTMENT_ID NAME                 NAME                      SALARY
------------- -------------------- ------------------------- --------
           90 Steven               King                      $24,000
           90 Neena                Kochhar                   $17,000
           90 Lex                  De Haan                   $17,000
           80 John                 Russell                   $14,000
           80 Karen                Partners                  $13,500
           20 Michael              Hartstein                 $13,000

6 rows selected.
```

Now reset PAGESIZE, NEWPAGE, and LINESIZE to their default values:

```
SET PAGESIZE 24
SET NEWPAGE 1
SET LINESIZE 80
```

To list the current values of these variables, use the SHOW command:

```
SHOW PAGESIZE
SHOW NEWPAGE
SHOW LINESIZE
```

Through the SQL*Plus command SPOOL, you can store your query results in a file or print them on your computer's default printer.

Storing and Printing Query Results

Send your query results to a file when you want to edit them with a word processor before printing or include them in a letter, memo, or other document.

To store the results of a query in a file—and still display them on the screen—enter the SPOOL command in the following form:

```
SPOOL file_name
```

If you do not follow the filename with a period and an extension, SPOOL adds a default file extension to the filename to identify it as an output file. The default varies with the host operating system; on most hosts it is LST or LIS. See the Oracle installation and user's manual(s) provided for your operating system for more information.

SQL*Plus continues to spool information to the file until you turn spooling off, using the following form of SPOOL:

```
SPOOL OFF
```

Creating a Flat File

When moving data between different software products, it is sometimes necessary to use a "flat" file (an operating system file with no escape characters, headings, or extra characters embedded). For example, if you do not have Oracle Net, you need to create a flat file for use with SQL*Loader when moving data from Oracle8 to Oracle9*i*.

To create a flat file with SQL*Plus, you first must enter the following SET commands:

```
SET NEWPAGE 0
SET SPACE 0
SET LINESIZE 80
SET PAGESIZE 0
SET ECHO OFF
SET FEEDBACK OFF
SET HEADING OFF
SET MARKUP HTML OFF SPOOL OFF
```

After entering these commands, you use the SPOOL command as shown in the previous section to create the flat file.

The SET COLSEP command may be useful to delineate the columns. For more information, see the SET command in Chapter 8.

Sending Results to a File

To store the results of a query in a file—and still display them on the screen—enter the SPOOL command in the following form:

```
SPOOL file_name
```

SQL*Plus stores all information displayed on the screen after you enter the SPOOL command in the file you specify.

Sending Results to a Printer

To print query results, spool them to a file as described in the previous section. Then, instead of using SPOOL OFF, enter the command in the following form:

```
SPOOL OUT
```

SQL*Plus stops spooling and copies the contents of the spooled file to your host computer's standard (default) printer. SPOOL OUT does not delete the spool file after printing.

Example 4–25 Sending Query Results to a Printer

To generate a final report and spool and print the results, create a command file named EMPRPT containing the following commands.

First, use EDIT to create the command file with your host operating system text editor. (Do not use INPUT and SAVE, or SQL*Plus will add a slash to the end of the file and will run the command file twice—once as a result of the semicolon and once due to the slash.)

```
EDIT EMPRPT
```

Next, enter the following commands into the file, using your text editor:

```
SPOOL TEMP
CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES

COLUMN DEPARTMENT_ID HEADING DEPARTMENT
COLUMN LAST_NAME HEADING 'LAST NAME'
COLUMN SALARY HEADING 'MONTHLY SALARY' FORMAT $99,999

BREAK ON DEPARTMENT_ID SKIP 1 ON REPORT
COMPUTE SUM OF SALARY ON DEPARTMENT_ID
COMPUTE SUM OF SALARY ON REPORT

SET PAGESIZE 24
SET NEWPAGE 0
SET LINESIZE 70

TTITLE CENTER 'A C M E  W I D G E T' SKIP 2 -
LEFT 'EMPLOYEE REPORT' RIGHT 'PAGE:' -
FORMAT 999 SQL.PNO SKIP 2
BTITLE CENTER 'COMPANY CONFIDENTIAL'

SELECT DEPARTMENT_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000
```

```
ORDER BY DEPARTMENT_ID;

SPOOL OFF
```

If you do not want to see the output on your screen, you can also add SET TERMOUT OFF to the beginning of the file and SET TERMOUT ON to the end of the file. Save and close the file in your text editor (you will automatically return to SQL*Plus). Now, run the command file EMPRPT:

```
@EMPRPT
```

SQL*Plus displays the output on your screen (unless you set TERMOUT to OFF), and spools it to the file TEMP:

```
                         A C M E  W I D G E T

EMPLOYEE REPORT                                              PAGE: 1

DEPARTMENT LAST NAME                 MONTHLY SALARY
---------- ------------------------- -------------
        20 Hartstein                        $13,000
********** ------------------------- -------------
sum                                         $13,000

        80 Russell                          $14,000
           Partners                         $13,500
********** ------------------------- -------------
sum                                         $27,500

        90 King                             $24,000
           Kochhar                          $17,000
           De Haan                          $17,000
********** ------------------------- -------------
sum                                         $58,000

                                    -------------
sum                                         $98,500
                       COMPANY CONFIDENTIAL

6 rows selected.
```

Creating Web Reports

SQL*Plus enables you to generate either a complete standalone web page, or HTML output which can be embedded in a web page. You can use SQLPLUS -MARKUP "HTML ON" or SET MARKUP HTML ON SPOOL ON to produce standalone web pages. SQL*Plus generates complete HTML pages automatically encapsulated with <HTML> and <BODY> tags.

The retrieved data is written to in HTML by default, though you can optionally direct output to the HTML <PRE> tag so that it displays in a web browser exactly as it appears in SQL*Plus. See the SQLPLUS -MARKUP command in the "Starting SQL*Plus Using the SQLPLUS Command" section of Chapter 7 and the SET MARKUP command in the SET section of Chapter 8 for more information about these commands.

SQLPLUS -MARKUP "HTML ON" is useful when embedding SQL*Plus in program scripts. On starting, it outputs the HTML and BODY tags before executing any commands. All subsequent output is in HTML until SQL*Plus terminates. The -SILENT and -RESTRICT command line options may be effectively used in conjunction with -MARKUP to suppress the display of SQL*Plus prompt and banner information and to restrict the use of some commands.

SET MARKUP HTML ON SPOOL ON generates complete HTML pages for each subsequently spooled file. The HTML tags in a spool file are closed when SPOOL OFF is executed or SQL*Plus exits.

You can use SET MARKUP HTML ON SPOOL OFF to generate HTML output suitable for embedding in an existing web page. HTML output generated this way has no <HTML> or <BODY> tags.

Creating Static Web Reports

You use the SET MARKUP command interactively during a SQL*Plus session to write HTML to a spool file. The output can be viewed in a web browser.

SET MARKUP only specifies that SQL*Plus output will be HTML encoded, it does not create or begin writing to an output file. You must use SET MARKUP HTML ON SPOOL ON to trigger the generation of the <HTML> and </HTML> tags when spool files are opened and closed. You then use the SQL*Plus SPOOL command to create and name a spool file, and to begin writing HMTL output to it.

When creating a HTML file, it is important and convenient to specify a *.html* file extension which is the standard file extension for HTML files. This allows you to easily identify the type of your output files, and also allows web browsers to

identify and correctly display your HTML files. If no extension is specified, the default SQL*Plus file extension is used.

You use SPOOL OFF or EXIT to append final HTML tags to the spool file and then close it. If you enter another SPOOL *filename* command, the current spool file is closed as for SPOOL OFF or EXIT, and a new HTML spool file with the specified name is created.

You can use the SET MARKUP command to enable or disable HTML output as required.

Example 4–26 Creating a Standalone Web Report in an Interactive Session

You can create HTML output in an interactive SQL*Plus session using the SET MARKUP command. You can include an embedded style sheet, or any other valid text in the HTML <HEAD> tag. Open a SQL*Plus session and enter the following:

```
SET MARKUP HTML ON SPOOL ON PREFORMAT OFF ENTMAP ON -
HEAD '<TITLE>Department Report</TITLE> -
<STYLE type="text/css"> -
<!-- BODY {background: #FFFFC6} --> -
</STYLE>' -
BODY 'TEXT="#FF00Ff"' -
TABLE 'WIDTH="90%" BORDER="5"'
```

You use the COLUMN command to control column output. The following COLUMN commands create new heading names for the SQL query output. The first command also turns off entity mapping for the DEPARTMENT_NAME column to allow HTML hyperlinks to be correctly created in this column of the output data:

```
COLUMN DEPARTMENT_NAME HEADING 'DEPARTMENT' ENTMAP OFF
COLUMN DEPARTMENT_NAME HEADING 'DEPARTMENT'
COLUMN CITY HEADING 'CITY'
```

SET MARKUP HTML ON SPOOL ON enables SQL*Plus to write HTML to a spool file. The following SPOOL command triggers the writing of the <HTML> and <BODY> tags to the named file:

```
SPOOL report.html
```

After the SPOOL command, anything entered or displayed on standard output is written to the spool file, *report.html*.

Enter a SQL query:

```
SELECT '<A HREF="http://oracle.com/'||DEPARTMENT_NAME||'.html">'||DEPARTMENT_
```

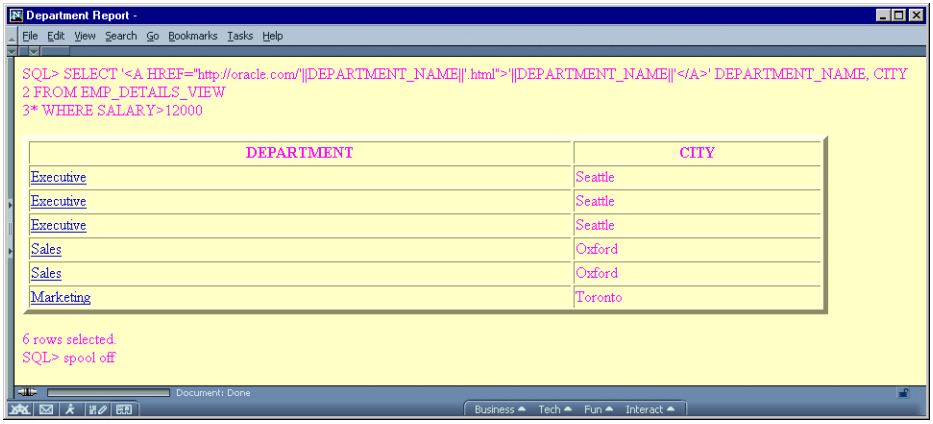
```
NAME||'</A>' DEPARTMENT_NAME, CITY
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000;
```

Enter the SPOOL OFF command:

```
SPOOL OFF
```

The </BODY> and </HTML> tags are appended to the spool file, *report.html*, before it is closed.

The output from report.sql is a file, *report.html*. This is a standalone web page that can be loaded into a web browser. Open *report.html* in your web browser. It should appear something like the following:



In this example, the prompts and query text have not been suppressed. Depending on how you invoke a script, you can use SET ECHO OFF of the -SILENT options to do this.

The SQL*Plus commands in this example contain several items of usage worth noting:

- The hyphen used to continue lines in long SQL*Plus commands.
- The TABLE option to set table WIDTH and BORDER attributes.
- The COLUMN command to set ENTMAP OFF for the DEPARTMENT_NAME column to enable the correct formation of HTML hyperlinks. This makes sure

that any HTML special characters such as quotes and angle brackets are not replaced by their equivalent entities, ", &, < and >.

- The use of quotes and concatenation characters in the SELECT statement to create hyperlinks by concatenating string and variable elements.

View the *report.html* source in your web browser, or in a text editor to see that the table cells for the Department column contain fully formed hyperlinks as shown:

```
<html>
<head>
<TITLE>Department Report</TITLE>  <STYLE type="text/css">  <!-- BODY
{background: #FFFFC6} -->  </STYLE>
<meta name="generator" content="SQL*Plus 9.0.1">
</head>
<body TEXT="#FF00Ff">
SQL&gt; SELECT '&lt;A HREF=&quot;http://oracle.com/'||DEPARTMENT_
NAME||'.html&quot;&gt;'||DEPARTMENT_NAME||'&lt;/A&gt;' DEPARTMENT_NAME, CITY
<br>
  2  FROM EMP_DETAILS_VIEW
<br>
  3* WHERE SALARY&gt;12000
<br>
<p>
<table WIDTH="90%" BORDER="5">
<tr><th>DEPARTMENT</th><th>CITY</th></tr>
<tr><td><A HREF="http://oracle.com/Executive.html">Executive</A></td>
<td>Seattle</td></tr>
<tr><td><A HREF="http://oracle.com/Executive.html">Executive</A></td>
<td>Seattle</td></tr>
<tr><td><A HREF="http://oracle.com/Executive.html">Executive</A></td>
<td>Seattle</td></tr>
<tr><td><A HREF="http://oracle.com/Sales.html">Sales</A></td>
<td>Oxford</td></tr>
<tr><td><A HREF="http://oracle.com/Sales.html">Sales</A></td>
<td>Oxford</td></tr>
<tr><td><A HREF="http://oracle.com/Marketing.html">Marketing</A></td>
<td>Toronto</td></tr>
</table>
<p>

6 rows selected.<br>


SQL&gt; spool off
<br>
```

```
</body>
</html>
```

Example 4–27 Creating a Standalone Web Report using the SQLPLUS command

Enter the following command at the operating system prompt:

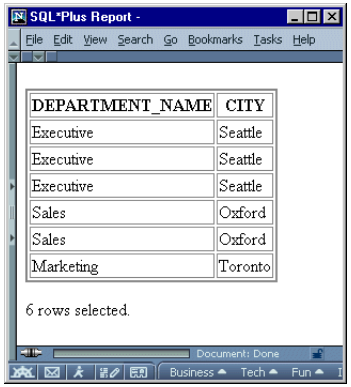
```
SQLPLUS -S -M "HTML ON TABLE 'BORDER="2"'" HR/HR@Ora9i @depart.sql>depart.html
```

where *depart.sql* contains:

```
SELECT DEPARTMENT_NAME, CITY
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000;
EXIT
```

This example starts SQL*Plus with user "HR", sets HTML ON, sets a BORDER attribute for TABLE, and runs the script *depart.sql*. The output from *departt.sql* is a complete web page which in this case has been redirected to the file *depart.html* using the > operating system command. It could be sent to a web browser if SQL*Plus was called in a web server CGI script. See "Creating a Web Report from a CGI Script" for information about calling SQL*Plus from a CGI script.

Start your web browser and enter the appropriate URL to open *depart.html*:



The SQLPLUS command in this example contains three layers of nested quotes. From the inside out, these are:

- "2" is a quoted HTML attribute value for BORDER.
- 'BORDER="2"' is the quoted text argument for the TABLE option.

- "HTML ON TABLE 'BORDER="2"'" is the quoted argument for the -MARKUP option.

The nesting of quotes may be different in some operating systems or program scripting languages.

Creating Dynamic Web Reports with CGI Scripts

The SQLPLUS -MARKUP command allows you to start a SQL*Plus session in Internet enabled mode, rather than using the SET MARKUP command interactively. This allows you to run a SQL*Plus session embedded inside a Common Gateway Interface (CGI) script or an operating system command file. A file created in this way can be displayed directly in a web browser. You can call SQL*Plus using any script language which is supported by your web server such as a UNIX shell script, a Windows command file (batch file), Java, JavaScript or a Perl file.

You can use this embedded approach to produce HTML web outputs that use existing SQL*Plus scripts unchanged. It provides an easy way to provide dynamically-created, web-based reports.

Example 4–28 Creating a Web Report from a CGI Script

You can use a CGI script to run SQL*Plus, and so produce a web report from a SQL script. There are three main elements required:

- A web page to call the CGI script.
- A CGI script to gather the input and run SQL*Plus.
- The SQL script to be run by the SQL*Plus session.

Web Page for CGI Example

In this example, the web page is a form which prompts for your username and password, a database connection string and the name of the SQL script to run.

Note: You need to carefully consider security on your server before embedding login information in a script file or using a CGI script to prompt for login information and pass it into the SQLPLUS command.

Consider setting initial conditions rather than assuming default values. For example, explicitly set ENTMAP ON even thought its default is ON.

```
<html>
<head><title>SQL*Plus CGI Report Demonstration</title></head>
<body bgcolor="#ffffff">

<h1>SQL*Plus CGI Report Demonstration</h1>

<!-- Change the URL here.  On Windows NT you may need to use
http://host.domain/cgi-bin/perl?plus.pl if your web server is not
configured to identify the script as a Perl program -->

<form method=post action="http://host.domain/cgi-bin/plus.pl">
<table border=0 summary="">

<tr>
  <td>Username:</td>
  <td><input type="text" name="username" size="10" align="left"></td>
</tr>
<tr>
  <td>Password:</td>
  <td><input type="password" name="password" size="10" align="left"></td>
</tr>
<tr>
  <td>Connect string: </td>
  <td><input type="text" name="db" size="10" align="left"></td>
</tr>
<tr>
  <td>Report to run: </td>
  <td><input type="text" name="script" value="employee.sql" size=40></td>
</tr>
<tr>
  <td><input type="submit" value="Run it"> <input type="reset"
value="Reset Form"></td>
  <td> </td>
</tr>
</table>
</form>
</body>
</html>
```

Perl Script for CGI Example

In this example, the CGI script is a Perl script, but it could be a shell script, a Java class or any other language supported by your web server. Create the following Perl CGI script and save it as *plus.pl* in the *cgi-bin* directory of your web server:

```
#! /usr/local/bin/perl -w
# Copyright (c) Oracle Corporation 1999, 2001. All Rights Reserved.
# NAME
#   plus.pl
# DESCRIPTION
#   This is a demonstration program to run a SQL*Plus report via CGI.
#   It is provided as is with no warranty implied or expressed.
#   Users are strongly recommended to understand error handling and
#   security issues before implementing CGI scripts.
#
# NOTES
#   This demostration requires that SQL*Plus 8.1.7 (or later) is
#   installed on your webserver, and the webserver is configured to
#   run CGI programs. The database may be on another machine, but
#   nust have Oracle Net access configured.
#
#   This demonstration consists of three files:
#      plus.html      - Sample HTML form that you open in your web
#                       browser. It calls plus.pl to run employee.sql
#      plus.pl        - Sample CGI program to run SQL*Plus
#      employee.sql   - Sample SQL script to generate a report from
#                       the HR sample schema.
#   These scripts need to be customized for your site.
#
# INSTALLATION INSTRUCTIONS:
#      1. Put plus.pl (this file) in the cgi-bin directory of your
#      web server and edit the environment variable section at the top
#      of the file.  Make the program executable, for example on UNIX,
#          chmod +x plus.pl
#      You may need to customize the top line of this script to point
#      to the Perl installation on your machine, and in the syntax
#      required for your operating system.
#      2. Put employee.sql in the cgi-bin directory too.
#      3. Put plus.html in a directory you can access from the web.
#      Edit plus.html to change the form URL to that of your web server.
#      4. Open plus.html in your browser and enter the fields. As
#      this demonstation uses the view, EMP_DETAILS_VIEW, from the HR
#      sample schema, enter the associated username, HR, and password.
#      If your database is not the default, or is on another machine,
#      enter a valid network alias, or full connection identifier in
#      the Connect Identifier field, otherwise leave it blank. If
#      employee.sql is in your cgi-bin directory,  you will probably
#      not need to specify a path, otherwise specify a machine path
#      and filename.
```

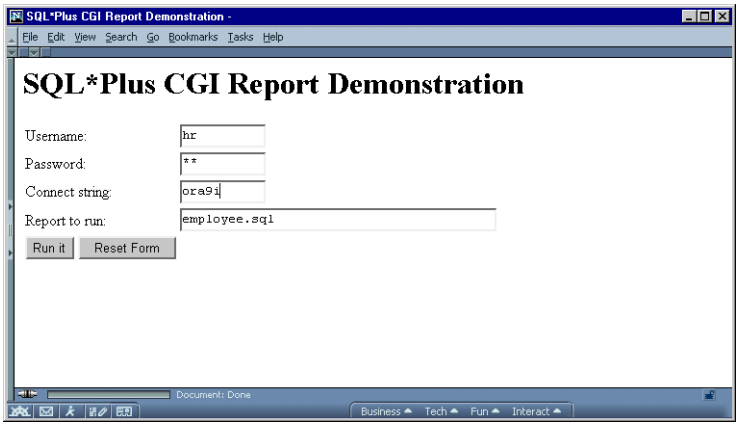
```
$debug = 0;   # Set this to 1 to see the form fields values entered.
# !!! Customize these environment variables and the executable name.
# !!! On Windows use "$ENV{'ORACLE_HOME'}\\bin\\sqlplus" for the executable.

# Set up the SQL*Plus environment
$ENV{'ORACLE_SID'} = "Ora9i";         # Your SID goes here
$ENV{'ORACLE_HOME'} = "/oracle/901";  # Your Oracle Home directory goes here
# $ENV{'TNS_ADMIN'} = "/var/opt/oracle";
$plusexe = "$ENV{'ORACLE_HOME'}/bin/sqlplus";

# Extract parameters and values from data entered through web browser
$i=<>;
@in = split(/[&;]/,$i);
foreach $i (0 .. $#in)
{ ($key,$val) = split(/=/,$in[$i],2);
# Change encoding to machine character set
   $key =~ s/%([A-Fa-f0-9]{2})/pack("c",hex($1))/ge;
   $val =~ s/%([A-Fa-f0-9]{2})/pack("c",hex($1))/ge;
# Store the value
   $in{"$key"} = $val;
}

# Construct the connection string from values passed in
$connstr = $in{'username'}."/".$in{'password'};
$connstr = $connstr."@".$in{'db'} if ($in{'db'});

# Construct the SQL script to be run
$script = "@".$in{'script'};

# Force output to be flushed
$| = 1;

# Print mime type
print "Content-Type: text/html\n\n";

if ($debug)
{ print "<html><body>\n";
  print "$plusexe:$connstr:$script:\n";
  print "</body></html>\n";
  exit;
}

# Call SQL*Plus with the parameters entered from the web browser
system ("$plusexe -r 3 -s -m \"html on\" $connstr $script");
exit;
```

SQL Script for CGI Example

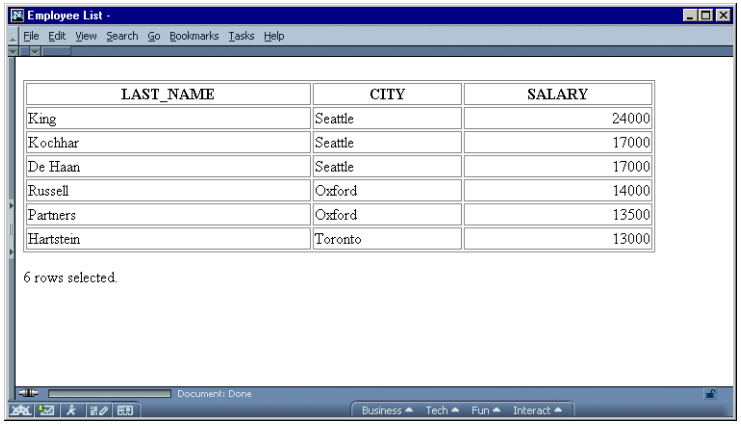
Create the following SQL*Plus script in a text editor and save it as *employee.sql* in the *cgi-bin* directory of your web server:

```
SELECT LAST_NAME, CITY, SALARY FROM EMP_DETAILS_VIEW;
EXIT;
```

Start your web browser and enter the appropriate URL to open *plus.html*:



Click **Run It** to execute the shell script *plus.pl*, which in turn starts SQL*Plus and runs the *employee.sql* script. The query results are displayed directly in your web browser:



Suppressing the Display of SQL*Plus Commands in Web Reports

It is recommended that you use SILENT mode to start your SQL*Plus session. This ensures that only the results of your SQL query appear in the web browser.

The SQLPLUS -SILENT option is particularly useful when used in combination with -MARKUP to generate embedded SQL*Plus reports using CGI scripts or operating system command files. It suppresses the display of SQL*Plus commands and the SQL*Plus banner. Your HTML output shows only the data resulting from your SQL query.

HTML Entities

Certain characters, <, >, " and & have a predefined meaning in HTML. In the previous example, you may have noticed that the > character was replaced by > as soon as you entered the SET MARKUP HTML ON command. To enable these characters to be displayed in your web browser, HTML provides character entities to use instead.

Table 4–2 Equivalent HTML Entities

Character	HTML Entity	Meaning
<	<	Start HTML tag label
>	>	End HTML tag label
"	"	Double quote
&	&	Ampersand

The web browser displays the > character, but the actual text in the HTML encoded file is the HTML entity, >. The SET MARKUP option, ENTMAP, controls the substitution of HTML entities. ENTMAP is set ON by default. It ensures that the characters <, >, " and & are always replaced by the HTML entities representing these characters. This prevents web browsers from misinterpreting these characters when they occur in your SQL*Plus commands, or in data resulting from your query.

You can set ENTMAP at a global level with SET MARKUP HTML ENTMAP ON, or at a column level with COLUMN *column_name* ENTMAP ON.

5

Database Administration

This chapter provides a brief overview of the database administration tools available in SQL*Plus, and discusses the following topics:

- Overview
- Introduction to Database Startup and Shutdown
- Redo Log Files
- Database Recovery

This chapter is intended for use by database administrators. In order to access the functionality of the commands mentioned in this chapter, database administrator privileges are necessary.

For more information on database administration, see the *Oracle9i Concepts* manual.

Overview

Special operations such as starting up or shutting down a database are performed by a database administrator (DBA). The DBA has certain privileges that are not assigned to normal users. The commands outlined in this chapter would normally be used by a DBA.

For more information about security and roles in SQL*Plus, see Appendix E, "Security".

Introduction to Database Startup and Shutdown

An Oracle database may not always be available to all users. To open or close a database, or to start up or shut down an instance, you must have dba privileges or be connected as SYSOPER or SYSDBA. Other users cannot change the current status of an Oracle database.

You cannot use STARTUP or SHUTDOWN to start or stop Oracle instances on Oracle7 servers.

Database Startup

Starting a database involves three steps:

1. **Starting an instance**

   An instance controls the background processes and the allocation of memory area to access an Oracle database.

2. **Mounting the database**

   Mounting the database associates it with a previously started instance.

3. **Opening the database**

   Opening the database makes it available for normal database operations.

For more information about database startup, see the *Oracle9i Concepts* guide. For information about the STARTUP command, see Chapter 8.

Example 5–1 Starting an Instance

To start an Oracle instance, without mounting the database, enter

```
STARTUP NOMOUNT
```

Example 5–2 Mounting the Database

To start an instance, mount the database, but leave the database closed, enter

```
STARTUP MOUNT
```

Example 5–3 Opening the Database

To start an instance using the Oracle9*i* Server parameter file INITSALE.ORA, mount and open the database named SALES in exclusive mode, and restrict access to administrative personnel, enter

```
STARTUP OPEN sales PFILE=INITSALE.ORA EXCLUSIVE RESTRICT
```

where SALES is the database name specified in the DB_NAME parameter in the INITSALE.ORA parameter file.

Database Shutdown

Shutting down a database involves three steps:

1. **Closing the database**

   When a database is closed, all database and recovery data in the SGA are written to the datafiles and redo log files, and closes all online datafiles.

2. **Dismounting the database**

   Dismounting the database disassociates the database from an instance and closes the control files of the database.

3. **Shutting down the instance**

   Shutting down an instance reclaims the SGA from memory and terminates the background Oracle processes that constitute an Oracle instance.

For more information about database shutdown, see the *Oracle9i Concepts* guide. For information about the SHUTDOWN command, see the "Command Reference" in Chapter 8.

Example 5–4 Shutting Down the Database

To shut down the database normally after it has been opened and mounted, enter

```
SHUTDOWN
Database closed.
Database dismounted.
ORACLE instance shut down.
```

Redo Log Files

Every Oracle database has a set of two or more redo log files. The set of redo log files for a database is collectively referred to as the database's *redo log*.

The redo log is used to record changes made to data. If, for example, there is a database failure, the redo log is used to recover the database. To protect against a failure involving the redo log itself, Oracle allows a *mirrored redo log* so that two or more copies of the redo log can be maintained on different disks.

ARCHIVELOG Mode

Operating a database in ARCHIVELOG mode enables the archiving of the online redo log.

The ARCHIVE LOG command permits a complete recovery from disk failure as well as instance failure, because all changes made to the database are permanently saved in an archived redo log.

For more information about redo log files and database archiving modes, see the *Oracle9i Concepts* manual. For information about using the ARCHIVE LOG command, see the "Command Reference" in Chapter 8.

To automatically begin archiving, enter

```
ARCHIVE LOG START
```

To list the details of the current log file being archived, enter

```
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
```

Database Recovery

If a damaged database is in ARCHIVELOG mode, it is a candidate for either complete media recovery or incomplete media recovery operations. To begin media recovery operations use the RECOVER command. For more information about using the RECOVER command, see the “Command Reference” in Chapter 8.

In order to begin recovery operations, you must have DBA privileges.

To recover the database up to a specified time using a control backup file, enter

```
RECOVER DATABASE UNTIL TIME '1998-11-23:12:47:30'-
USING BACKUP CONTROLFILE
```

To recover two offline table-spaces, enter

```
RECOVER TABLESPACE ts1, ts2
```

Make sure that the table-spaces you are interested in recovering have been taken offline, before proceeding with recovery for those table-spaces.

6

Accessing SQL Databases

This chapter explains how to access databases through SQL*Plus, and discusses the following topics:

- Connecting to the Default Database
- Connecting to a Remote Database
- Copying Data from One Database to Another
- Copying Data between Tables on One Database

Read this chapter while sitting at your computer and try out the example shown. Before beginning, make sure you have access to the sample tables described in Chapter 1.

Connecting to the Default Database

To access data in a given database, you must first connect to the database. When you start SQL*Plus, you normally connect to your default Oracle database under the username and password you enter while starting. Once you have logged in, you can connect under a different username with the SQL*Plus CONNECT command. The username and password must be valid for the database. See Username and Password in Chapter 1 for a list of default user logins created during installation.

For example, to connect the username TODD to the default database using the password FOX, you could enter

```
CONNECT TODD/FOX
```

If you omit the username and password, SQL*Plus prompts you for them. You also have the option of typing only the username following CONNECT and omitting the password (SQL*Plus then prompts for the password). Because CONNECT first disconnects you from your current database, you will be left unconnected to any database if you use an invalid username and password in your CONNECT command.

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 as this user is not permitted until the account is unlocked by your DBA.

You can disconnect the username currently connected to Oracle without leaving SQL*Plus by entering the SQL*Plus command DISCONNECT at the SQL*Plus command prompt.

The default database is configured at an operating system level by setting operating system environment variables, symbols or, possibly, by editing an Oracle specific configuration file. Refer to your Oracle documentation for your operating system for more information.

Connecting to a Remote Database

Many large installations run Oracle on more than one computer. Such computers are often connected in a network, which permits programs on different computers to exchange data rapidly and efficiently. Networked computers can be physically near each other, or can be separated by large distances and connected by telecommunication links.

Databases on other computers or databases on your host computer other than your default database are called *remote databases*. You can access remote databases if the desired database has Oracle Net and both databases have compatible network drivers.

You can connect to a remote database in one of two ways:

- From within SQL*Plus, using the CONNECT command.
- As you start SQL*Plus, using the SQLPLUS command.

Connecting to a Remote Database from within SQL*Plus

To connect to a remote database using CONNECT, include a Oracle Net database specification in the CONNECT command in one of the following forms (the username and password you enter must be valid for the database to which you wish to connect):

- CONNECT HR@*connect_identifier*
- CONNECT HR/HR@*connect_identifier*

SQL*Plus prompts you for a password as needed, and connects you to the specified database.

Like any database connection, 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 as this user is not permitted until the account is unlocked by your DBA.

When you connect to a remote database in this manner, you can use the complete range of SQL and SQL*Plus commands and PL/SQL blocks on the database.

The exact string you enter for the service name depends upon the Oracle Net protocol your computer uses. For more information, see CONNECT in Chapter 8 and the Oracle Net guide appropriate for your protocol, or contact your DBA.

Connecting to a Remote Database as You Start SQL*Plus

To connect to a remote database when you start SQL*Plus, include the Oracle Net service name in your SQLPLUS command in one of the following forms:

- SQLPLUS HR*@connect_identifier*
- SQLPLUS HR/ HR*@connect_identifier*

You must use a username and password valid for the remote database and substitute the appropriate service name for the remote database. SQL*Plus prompts you for username and password as needed, starts SQL*Plus, and connects you to the specified database. This is the database used until you CONNECT to another database, DISCONNECT, or leave SQL*Plus.

Like any database connection, 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 as this user is not permitted until the account is unlocked by your DBA.

Once again, you can manipulate tables in the remote database directly after you connect in this manner.

Note: Do not confuse the @ symbol of the connect identifier with the @ command used to run a command file.

Copying Data from One Database to Another

Use the SQL*Plus COPY command to copy CHAR, DATE, LONG, NUMBER or VARCHAR2 data between databases and between tables on the same database. With the COPY command, you can copy data between databases in the following ways:

- Copy data from a remote database to your local database.
- Copy data from your local (default) database to a remote database (most systems).
- Copy data from one remote database to another remote database (most systems).

Note: In general, the COPY command was designed to be used for copying data between Oracle and non-Oracle databases. You should use SQL commands (CREATE TABLE AS and INSERT) to copy data between Oracle databases.

Understanding COPY Command Syntax

You enter the COPY command in the following form:

```
COPY FROM database TO database action -
destination_table (column_name, column_name, -
column_name ...) USING query
```

Here is a sample COPY command:

```
COPY FROM HR/HR@BOSTONDB -
TO TODD/FOX@CHICAGODB -
CREATE NEWDEPT (DEPARTMENT_ID, DEPARTMENT_NAME, CITY) -
USING SELECT * FROM EMP_DETAILS_VIEW
```

To specify a database in the FROM or TO clause, you must have a valid username and password for the local and remote database(s) and know the appropriate Oracle Net service name(s). COPY obeys Oracle security, so the username you specify must have been granted access to tables for you to have access to tables. For information on what databases are available to you, contact your DBA.

When you copy to your local database from a remote database, you can omit the TO clause. When you copy to a remote database from your local database, you can omit the FROM clause. When you copy between remote databases, you must include both clauses. However, including both clauses benefits the readability of your scripts.

The COPY command behaves differently based on whether the destination table already exists and on the action clause you enter (CREATE in the example above). For more information, see the section "Controlling Treatment of the Destination Table" later in this chapter.

By default, the copied columns have the same names in the destination table that they have in the source table. If you want to give new names to the columns in the destination table, enter the new names in parentheses after the destination table name. If you enter any column names, you must enter a name for every column you are copying.

> **Note:** 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 USING clause specifies a query that names the source table and specifies the data that COPY copies to the destination table. You can use any form of the SQL SELECT command to select the data that the COPY command copies.

Here is an example of a COPY command that copies only two columns from the source table, and copies only those rows in which the value of DEPARTMENT_ID is 30:

```
COPY FROM HR/HR@BOSTONDB -
REPLACE EMPCOPY2 -
USING SELECT LAST_NAME, SALARY -
FROM EMP_DETAILS_VIEW -
WHERE DEPARTMENT_ID = 30
```

You may find it easier to enter and edit long COPY commands in command files rather than trying to enter them directly at the command prompt.

Controlling Treatment of the Destination Table

You control the treatment of the destination table by entering one of four control clauses—REPLACE, CREATE, INSERT, or APPEND.

The REPLACE clause names the table to be created in the destination database and specifies the following actions:

- If the destination table already exists, COPY drops the existing table and replaces it with a table containing the copied data.
- If the destination table does not already exist, COPY creates it using the copied data.

You can use the CREATE clause to avoid accidentally writing over an existing table. CREATE specifies the following actions:

- If the destination table already exists, COPY reports an error and stops.

- If the destination table does not already exist, COPY creates the table using the copied data.

Use INSERT to insert data into an existing table. INSERT specifies the following actions:

- If the destination table already exists, COPY inserts the copied data in the destination table.
- If the destination table does not already exist, COPY reports an error and stops.

Use APPEND when you want to insert data in an existing table, or create a new table if the destination table does not exist. APPEND specifies the following actions:

- If the destination table already exists, COPY inserts the copied data in the destination table.
- If the table does not already exist, COPY creates the table and then inserts the copied data in it.

Example 6–1 Copying from a Remote Database to Your Local Database Using CREATE

To copy HR from a remote database into a table called EMPLOYEE_COPY on your own database, enter the following command:

> **Note:** See your DBA for an appropriate username, password, and service name for a remote computer that contains a copy of EMPLOYEE_COPY.

```
COPY FROM HR/HR@BOSTONDB -
CREATE EMPCOPY -
USING SELECT * FROM HR
```

```
Array fetch/bind size is 15. (arraysize is 15)
Will commit when done. (copycommit is 0)
Maximum long size is 80. (long is 80)
```

SQL*Plus then creates the table EMPLOYEE_COPY and copies the rows:

```
Table SALESMAN created.

   5 rows selected from HR@BOSTONDB.
   5 rows inserted into SALESMAN.
   5 rows committed into SALESMAN at DEFAULT HOST connection.
```

In this COPY command, the FROM clause directs COPY to connect you to the database with the specification D:BOSTON-MFG as HR, with the password HR.

Notice that you do not need a semicolon at the end of the command; COPY is a SQL*Plus command, not a SQL command, even though it contains a query. Since most COPY commands are longer than one line, you must use a hyphen (-), optionally preceded by a space, at the end of each line except the last.

Interpreting the Messages that COPY Displays

The first three messages displayed by COPY show the values of SET command variables that affect the COPY operation. The most important one is LONG, which limits the length of a LONG column's value. (LONG is a datatype, similar to CHAR.) If the source table contains a LONG column, COPY truncates values in that column to the length specified by the system variable LONG.

The variable ARRAYSIZE limits the number of rows that SQL*Plus fetches from the database at one time. This number of rows makes up a *batch*. The variable COPYCOMMIT sets the number of batches after which COPY commits changes to the database. (If you set COPYCOMMIT to zero, COPY commits changes only after all batches are copied.) For more information on the variables of the SET command, including how to change their settings, see the SET command in Chapter 8.

After listing the three system variables and their values, COPY tells you if a table was dropped, created, or updated during the copy. Then COPY lists the number of rows selected, inserted, and committed.

Specifying Another User's Table

You can refer to another user's table in a COPY command by qualifying the table name with the username, just as you would in your local database, or in a query with a database link.

For example, to make a local copy of a table named DEPARTMENT owned by the username ADAMS on the database associated with the Oracle Net connect identifier BOSTONDB, you would enter

```
COPY FROM HR/HR@BOSTONDB -
CREATE EMPLOYEE_COPY2 -
USING SELECT * FROM ADAMS.DEPT
```

Of course, you could get the same result by instructing COPY to log in to the remote database as ADAMS. You cannot do that, however, unless you know the password associated with the username ADAMS.

Copying Data between Tables on One Database

You can copy data from one table to another in a single database (local or remote). To copy between tables in your local database, specify your own username and password and the service name for your local database in either a FROM or a TO clause (omit the other clause):

```
COPY FROM HR/HR@MYDATABASE -
INSERT EMPLOYEE_COPY2 -
USING SELECT * FROM EMPLOYEE_COPY
```

To copy between tables on a remote database, include the same username, password, and service name in the FROM and TO clauses:

```
COPY FROM HR/HR@BOSTONDB -
TO HR/HR@BOSTONDB -
INSERT EMPLOYEE_COPY2 -
USING SELECT * FROM EMPLOYEE_COPY
```

Part II

Reference

This section provides an overview of how to start SQL*Plus. It also provides a Command Reference that contains a description of each SQL*Plus command.

The following chapters and appendices are covered in this section:

- Starting SQL*Plus and Getting Help
- Command Reference

7

Starting SQL*Plus and Getting Help

This chapter explains how to access SQL*Plus from the operating system prompt, and discusses the following topics:

- Starting SQL*Plus Using the SQLPLUS Command
- Getting Help

Starting SQL*Plus Using the SQLPLUS Command

You use the SQLPLUS command at the operating system prompt to start SQL*Plus:

```
SQLPLUS [ [Options] [Logon] [Start] ]
```

where:

Options has the following syntax:

```
 -H[ELP] | -V[ERSION]
| [ [-M[ARKUP] markup_option] [-R[ESTRICT] {1|2|3}] [-S[ILENT]] ]
```

and *markup_option* has the following syntax:

```
 HTML [ON|OFF] [HEAD text] [BODY text] [TABLE text]
     [ENTMAP {ON|OFF}] [SPOOL {ON|OFF}]  [PRE[FORMAT] {ON|OFF}]
```

Logon has the following syntax:

```
{username[/password][@connect_identifier | / }
[AS {SYSOPER|SYSDBA}] | /NOLOG
```

Start has the following syntax:

```
@{uri|file_name[.ext]} [arg ...]
```

You have the option of entering *logon*. If you do not specify *logon* and do specify *start*, SQL*Plus assumes that the first line of the command file contains a valid logon. If neither *start* nor *logon* are specified, SQL*Plus prompts for logon information.

The following sections contain descriptions of SQLPLUS command terms:

Options

HELP Option

-H[ELP]

Displays the usage and syntax for the SQLPLUS command, and then returns control to the operating system.

VERSION Option

-V[ERSION]

Displays the current version and level number for SQL*Plus, and then returns control to the operating system.

MARKUP Options

-M[ARKUP]

You can use the MARKUP option to generate a complete stand alone web page from your query or script. MARKUP currently supports HTML 4.0 transitional.

Use SQLPLUS -MARKUP HTML ON or SET MARKUP HTML ON SPOOL ON to produce standalone web pages. SQL*Plus will generate complete HTML pages automatically encapsulated with <HTML> and <BODY> tags. The HTML tags in a spool file are closed when SPOOL OFF is executed or SQL*Plus exits.

The -SILENT and -RESTRICT command line options may be useful when used in conjunction with -MARKUP.

You can use SET MARKUP HTML ON SPOOL OFF to generate HTML output suitable for embedding in an existing web page. Output generated this way has no <HTML> or <BODY> tags.

In this release, you can use MARKUP HTML ON to produce HTML output in either the <PRE> tag or in an HTML table. Output to a table uses standard HTML <TABLE>, <TR> and <TD> tags to automatically encode the rows and columns resulting from a query. Output to an HTML table is now the default behavior when the HTML option is set ON. You can generate output using HTML <PRE> tags by setting PREFORMAT ON.

Use the SHOW MARKUP command to view the status of MARKUP options.

The SQLPLUS -MARKUP command has the same options and functionality as the SET MARKUP command. These options are described in this section. For other information on the SET MARKUP command, see the SET command in Chapter 8.

Note: Depending on your operating system, the complete *markup_option* clause for the SQLPLUS command may need to be contained in quotes.

HTML [ON|OFF]

HTML is a mandatory MARKUP argument which specifies that the type of output to be generated is HTML. The optional HTML arguments, ON and OFF, specify whether or not to generate HTML output. The default is OFF.

MARKUP HTML ON generates HTML output using the specified MARKUP options, or in the case of SET MARKUP, options set by previous SET MARKUP HTML commands in the current session.

You can turn HTML output ON and OFF as required during a session. The default is OFF.

You enable the writing of HTML output with the MARKUP option, SPOOL ON, and you subsequently initiate writing output to a spool file with the SQL*Plus command, SPOOL *filename*. See SPOOL {ON | OFF} below, and the SPOOL command in Chapter 8 for more information.

HEAD *text*

The HEAD *text* option allows you to specify content for the <HEAD> tag. By default, *text* is:

```
<TITLE>SQL*Plus Report</TITLE>
```

If text includes spaces, it must be enclosed in quotes. SQL*Plus does not test this free text entry for HTML validity. You must ensure that the text you enter is valid for the HTML <HEAD> tag. This gives you the flexibility to customize output for your browser or special needs.

BODY *text*

The BODY *text* option allows you to specify attributes for the <BODY> tag. By default, there are no attributes. If text includes spaces, it must be enclosed in quotes. SQL*Plus does not test this free text entry for HTML validity. You must ensure that the text you enter is valid for the HTML <BODY> tag. This gives you the flexibility to customize output for your browser or special needs.

TABLE *text*

The TABLE *text* option allows you to enter attributes for the <TABLE> tag. You can use TABLE *text* to set HTML <TABLE> tag attributes such as BORDER, CELLPADDING, CELLSPACING and WIDTH. By default, the <TABLE> WIDTH attribute is set to 90% and the BORDER attribute is set to 1.

If *text* includes spaces, it must be enclosed in quotes. SQL*Plus does not test this free text entry for HTML validity. You must ensure that the text you enter is valid for the HTML <TABLE> tag. This gives you the flexibility to customize output for your browser or special needs.

ENTMAP {ON|OFF}

ENTMAP ON or OFF specifies whether or not SQL*Plus replaces special characters <, >, " and & with the HTML entities <, > " and & respectively. ENTMAP is set ON by default.

You can turn ENTMAP ON and OFF as required during a session. For example, with ENTMAP OFF, SQL*Plus screen output is:

```
SQL>SELECT DEPARTMENT_ID, CITY
  1  FROM EMP_DETAILS_VIEW
  2  WHERE SALARY = 12000;
```

With ENTMAP ON, SQL*Plus screen output is:

```
SQL&gt; SELECT DEPARTMENT_ID, CITY
  2  FROM EMP_DETAILS_VIEW
  3  WHERE SALARY = 12000;
```

If entities are not mapped, web browsers may treat data as invalid HTML and all subsequent output may display incorrectly. ENTMAP OFF allows users to write their own HTML tags to customize output.

As entities in the <HEAD> and <BODY> tags are not mapped, you must ensure that valid entities are used in the MARKUP HEAD and BODY options.

Note: ENTMAP only has affect when either the HTML option is set ON, or the SPOOL option is set ON. For more information about using entities in your output, see the COLUMN command in Chapter 8.

SPOOL {ON|OFF}

SPOOL ON or OFF specifies whether or not SQL*Plus writes the HTML opening tags, <HTML> and <BODY>, and the closing tags, </BODY> and </HTML>, to the start and end of each file created by the SQL*Plus SPOOL *filename* command. The default is OFF.

You can turn SPOOL ON and OFF as required during a session.

Note: It is important to distinguish between the SET MARKUP HTML SPOOL option, and the SQLPLUS SPOOL *filename* command.

The SET MARKUP HTML SPOOL ON option enables the writing of the <HTML> tag to the spool file. The spool file is not created, and the header and footer tags enabled by the SET MARKUP HTML SPOOL ON option are not written to the spool file until you issue the SQLPLUS SPOOL *filename* command.

SQL*Plus writes several HTML tags to the spool file when you issue the SPOOL *filename* command. The tags written and their default content are:

```
<HTML>
<HEAD>
<TITLE>SQL*Plus Report</TITLE>
<META name="generator" content="SQL*Plus 9.0.1">
</HEAD>
<BODY>
```

When you issue any of the SQL*Plus commands: EXIT, SPOOL OFF or SPOOL *filename*, SQL*Plus appends the following end tags and closes the file:

```
</BODY>
</HTML>
```

You can specify <HEAD> tag contents and <BODY> attributes using the HEAD and BODY options

PRE[FORMAT] {ON|OFF}

PREFORMAT ON or OFF specifies whether or not SQL*Plus writes output to the <PRE> tag or to an HTML table. The default is OFF, so output is written to a HTML table by default. You can turn PREFORMAT ON and OFF as required during a session.

Notes: To produce report output using the HTML <PRE> tag, you must set PREFORMAT ON. For example:

```
SQLPLUS -M "HTML ON PREFORMAT ON"
SET MARKUP HTML ON PREFORMAT ON
```

MARKUP Usage Notes

Existing scripts that do not explicitly set PREFORMAT ON will generate output in HTML tables. If you want output in HTML <PRE> tags, you must set PREFORMAT ON.

Some SQL*Plus commands have different behavior when output is directed to an HTML table. Commands originally intended to format paper reports may have different meaning for reports intended for web tables:

- PAGESIZE is the number of rows in an HTML table, not the number of lines. Each row may contain multiple lines. The TTITLE, BTITLE and column headings are repeated every PAGESIZE rows.
- LINESIZE may have an effect on data if wrapping is on, or for very long data. Depending on data size, they may be generated on separate lines, which a browser may interpret as a space character.
- TTITLE and BTITLE content is output to three line positions: left, center and right, and the maximum line width is preset to 90% of the browser window. These elements may not align with the main output as expected due to the way they are handled for web output. Entity mapping in TTITLE and BTITLE is the same as the general ENTMAP setting specified in the MARKUP command.
- If you use a title in your output, then SQL*Plus starts a new HTML table for output rows that appear after the title. Your browser may format column widths of each table differently, depending on the width of data in each column.
- SET COLSEP and RECSEP only produce output in HTML reports when PREFORMAT is ON.

RESTRICT Option

-R[ESTRICT] {1|2|3}

Allows you to disable certain commands that interact with the operating system. This is similar to disabling the same commands in the Product User Profile (PUP) table. However, commands disabled with the

-RESTRICT option are disabled even if there is no connection to a server, and remain disabled until SQL*Plus terminates.

If no -RESTRICT option is active, than all commands can be used, unless disabled in the PUP table.

If -RESTRICT 3 is used, then LOGIN.SQL is not read. GLOGIN.SQL is read but restricted commands used will fail.

Table 7-1 shows the commands disabled in each restriction level.

Table 7–1 Commands Disabled by Restriction Level

Command	Level 1	Level 2	Level 3
EDIT	disabled	disabled	disabled
GET			disabled
HOST, !	disabled	disabled	disabled
SAVE		disabled	disabled
SPOOL		disabled	disabled
START, @, @@			disabled
STORE		disabled	disabled

SILENT Option

-S[ILENT]

Suppresses all SQL*Plus information and prompt messages, including the command prompt, the echoing of commands, and the banner normally displayed when you start SQL*Plus. If you omit *username* or *password*, SQL*Plus prompts for them, but the prompts are not visible. Use SILENT to invoke SQL*Plus within another program so that the use of SQL*Plus is invisible to the user.

SILENT is a useful mode for creating reports for the web using the SQLPLUS -MARKUP command inside a CGI script or operating system command file. The SQL*Plus banner and prompts are suppressed and do not appear in reports created using the SILENT option.

Logon

username[*/password*]

Represent the username and password with which you wish to start SQL*Plus and connect to Oracle. If you omit *username* and *password*, SQL*Plus prompts you for them.

If you omit only *password*, SQL*Plus prompts you for *password*. When prompting, SQL*Plus does not display *password* on your terminal screen. In silent mode, username and password prompts are not visible – your username will appear as you type it, but not your password.

@connect_identifier

Consists of an Oracle Net connect identifier. The exact syntax depends upon the Oracle Net communications protocol your Oracle installation uses. For more information, refer to the Oracle Net manual appropriate for your protocol or contact your DBA.

/

Represents a default logon using operating system authentication. You cannot enter a *connect_identifer* if you use a default logon. In a default logon, SQL*Plus typically attempts to log you in using the username OPS$*name*, where *name* is your operating system username. Note that the prefix "OPS$" can be set to any other string of text. For example, you may wish to change the settings in your INIT.ORA parameters file to LOGON*name* or USERID*name*. See the *Oracle9i 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. You can also use either of these privileged connections with / and /NOLOG.

If you use this option, you need to quote the command arguments on many operating systems, for example:

```
SQLPLUS "/ AS SYSDBA"
SQLPLUS "SYSTEM/MANAGER AS SYSOPER"
```

/NOLOG

Establishes no initial connection to Oracle. Before issuing any SQL commands, you must issue a CONNECT command to establish a valid

logon. Use /NOLOG when you want to have a SQL*Plus command file prompt for the username, password, or database specification. The first line of this command file is not assumed to contain a logon.

Start

@{uri|file_name[*.ext*]} [*arg* ...]

Specifies the name of a command file and arguments to run. The command file can be called from the local file system or from a web server. *uri* is only supported on Windows platforms in this release.

SQL*Plus passes the arguments to the command file as if executing the file using the SQL*Plus START command. If no file suffix (file extension) is specified, the suffix defined by the SET SUFFIX command is used. The default suffix is *.sql*.

See the START command in Chapter 8 for more information.

Setting Up the Site Profile

SQL*Plus supports a Site Profile, a SQL*Plus command file created by the database administrator. This file is generally named GLOGIN with an extension of SQL. SQL*Plus executes this command file whenever any user starts SQL*Plus and SQL*Plus establishes the Oracle connection. The Site Profile allows the DBA to set up SQL*Plus environment defaults for all users at a particular site; users cannot directly access the Site Profile. The default name and location of the Site Profile depend on your system. Site Profiles are described in more detail in the Oracle installation and user's manual(s) provided for your operating system.

Setting Up the User Profile

SQL*Plus also supports a User Profile, executed after the Site Profile. SQL*Plus searches for a file named LOGIN with the extension SQL in your current directory. If SQL*Plus does not find the file there, SQL*Plus will search a system-dependent path to find the file. Some operating systems may not support this path search.

Receiving a Return Code

If you fail to log in successfully to SQL*Plus because your username or password is invalid or some other error, SQL*Plus will return an error status equivalent to an EXIT FAILURE command. See the EXIT command in this chapter for further information.

Example 7–1 Starting SQL*Plus

To start SQL*Plus with *username* HR and *password* HR, enter

```
SQLPLUS HR/HR
```

To start SQL*Plus, as above, and to make POLICY the default database (where POLICY is a valid Oracle Net database connect identifier), enter

```
SQLPLUS HR/HR@POLICY
```

To start SQL*Plus with *username* HR and *password* HR and run a command file named STARTUP with the extension SQL, enter

```
SQLPLUS HR/HR @STARTUP
```

Note the space between HR and @STARTUP.

To start SQL*Plus with HTML ON, so that output can be captured in a file and then viewed on a web browser, enter

```
SQLPLUS -M "HTML ON" HR/HR
```

To start SQL*Plus with no access to the EDIT and HOST commands during the session, enter

```
SQLPLUS -R 1 HR/HR
```

Example 7–2 Displaying the SQLPLUS syntax

To display the syntax of the SQLPLUS command, enter

```
SQLPLUS -H
```

```
Usage: SQLPLUS [ [<option>] [<logon>] [<start>] ]
where <option> ::= -H | -V | [ [-M <o>] [-R <n>] [-S] ]
      <logon>  ::= <username>[/<password>][@<connect_identifier>] | / |/NOLOG
      <start>  ::= @<uri>|<filename>[.<ext>] [<parameter> ...]
        -H displays the SQL*Plus version banner and usage syntax
        -V displays the SQL*Plus version banner
        -M <o> uses HTML markup options <o>
        -R <n> uses restricted mode <n>
        -S uses silent mode
```

Getting Help

To access online help for SQL*Plus commands, you can type HELP followed by the command name at the SQL command prompt. For example:

```
HELP ACCEPT
```

To display a list of SQL*Plus commands, type HELP followed by either TOPICS or INDEX. HELP TOPICS displays a single column list of SQL*Plus commands. HELP INDEX displays a four column list of SQL*Plus commands which usually fits in a single screen. For example:

```
HELP INDEX
```

If you get a response that help is unavailable, consult your database administrator. See the HELP command in Chapter 8 for more information.

8

Command Reference

This chapter contains descriptions of SQL*Plus commands, listed alphabetically. Use this chapter for reference only. Each description contains the following parts:

Syntax	Shows how to enter the command and provides a brief description of the basic uses of the command. Refer to "Conventions in Code Examples" in the Preface for an explanation of the syntax notation
Terms	Describes the function of each term or clause appearing in the syntax.
Usage	Provides additional information on uses of the command and on how the command works.
Examples	Gives one or more examples of the command.

A summary table that lists and briefly describes SQL*Plus commands precedes the individual command descriptions.

You can continue a long SQL*Plus command by typing a hyphen at the end of the line and pressing [Return]. If you wish, you can type a space before typing the hyphen. SQL*Plus displays a right angle-bracket (>) as a prompt for each additional line.

You do not need to end a SQL*Plus command with a semicolon. When you finish entering the command, you can just press [Return]. If you wish, however, you can enter a semicolon at the end of a SQL*Plus command.

SQL*Plus Command Summary

Command	Page	Description
@	8-5	Runs the SQL*Plus statements in the specified command file. The command file can be called from the local file system or from a web server.
@@	8-7	Runs a command file. This command is identical to the @ ("at" sign) command. It is useful for running nested command files because it looks for the specified command file in the same path as the command file from which it was called.
/	8-9	Executes the SQL command or PL/SQL block.
ACCEPT	8-10	Reads a line of input and stores it in a given user variable.
APPEND	8-12	Adds specified text to the end of the current line in the buffer.
ARCHIVE LOG	8-13	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	8-16	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	8-18	Specifies where and how formatting will change in a report, or lists the current break definition.
BTITLE	8-23	Places and formats a specified title at the bottom of each report page, or lists the current BTITLE definition.
CHANGE	8-24	Changes text on the current line in the buffer.
CLEAR	8-27	Resets or erases the current clause or setting for the specified option, such as BREAKS or COLUMNS.
COLUMN	8-29	Specifies display characteristics for a given column, or lists the current display characteristics for a single column or for all columns.
COMPUTE	8-40	Calculates and prints summary lines, using various standard computations, on subsets of selected rows, or lists all COMPUTE definitions.
CONNECT	8-46	Connects a given user to Oracle.
COPY	8-48	Copies results from a query to a table in a local or remote database.

Command	Page	Description
DEFINE	8-52	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	8-54	Deletes one or more lines of the buffer.
DESCRIBE	8-56	Lists the column definitions for the specified table, view, or synonym or the specifications for the specified function or procedure.
DISCONNECT	8-62	Commits pending changes to the database and logs the current user off Oracle, but does not exit SQL*Plus.
EDIT	8-63	Invokes a host operating system text editor on the contents of the specified file or on the contents of the buffer.
EXECUTE	8-65	Executes a single PL/SQL statement.
EXIT	8-66	Terminates SQL*Plus and returns control to the operating system.
GET	8-68	Loads a host operating system file into the buffer.
HELP	8-69	Accesses the SQL*Plus help system.
HOST	8-70	Executes a host operating system command without leaving SQL*Plus.
INPUT	8-72	Adds one or more new lines after the current line in the buffer.
LIST	8-74	Lists one or more lines of the buffer.
PASSWORD	8-76	Allows a password to be changed without echoing the password on an input device.
PAUSE	8-77	Displays the specified text, then waits for the user to press [Return].
PRINT	8-78	Displays the current value of a bind variable.
PROMPT	8-79	Sends the specified message to the user's screen.
QUIT	8-66	Terminates SQL*Plus and returns control to the operating system. QUIT is identical to EXIT.
RECOVER	8-80	Performs media recovery on one or more tablespaces, one or more datafiles, or the entire database.
REMARK	8-86	Begins a comment in a command file.
REPFOOTER	8-87	Places and formats a specified report footer at the bottom of each report, or lists the current REPFOOTER definition.

Command	Page	Description
REPHEADER	8-89	Places and formats a specified report header at the top of each report, or lists the current REPHEADER definition.
RUN	8-93	Lists and runs the SQL command or PL/SQL block currently stored in the buffer.
SAVE	8-94	Saves the contents of the buffer in a host operating system file (a command file).
SET	8-96	Sets a system variable to alter the SQL*Plus environment for your current session.
SHOW	8-122	Shows the value of a SQL*Plus system variable or the current SQL*Plus environment.
SHUTDOWN	8-127	Shuts down a currently running Oracle instance.
SPOOL	8-129	Stores query results in an operating system file and, optionally, sends the file to a printer.
START	8-130	Runs the SQL*Plus statements in the specified command file. The command file can be called from the local file system or from a web server.
STARTUP	8-132	Starts an Oracle instance and optionally mounts and opens a database.
STORE	8-135	Saves attributes of the current SQL*Plus environment in a host operating system file (a command file).
TIMING	8-136	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	8-138	Places and formats a specified title at the top of each report page, or lists the current TTITLE definition.
UNDEFINE	8-142	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	8-143	Declares a bind variable that can be referenced in PL/SQL.
WHENEVER OSERROR	8-150	Exits SQL*Plus if an operating system command generates an error.
WHENEVER SQLERROR	8-152	Exits SQL*Plus if a SQL command or PL/SQL block generates an error.

@ (“at” sign)

Syntax

@{*uri*|*file_name*[.*ext*] } [*arg...*]

Runs the SQL*Plus statements in the specified command file. The command file can be called from the local file system or from a web server. *uri* is only supported on Windows platforms in this release.

Terms

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

uri

Specifies the Uniform Resource Identifier of a script to run on the specified web server. SQL*Plus supports HTTP, FTP and gopher protocols.

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. See 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

In a command file, you can include 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, "Security"), 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. If you require a semicolon in your command, add a second 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

```
@PRINTRPT
```

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

```
@WKRPT.QRY
```

You can run a script named YEAREND specified by a Uniform Resource Indentifier, and pass values to variables referenced in YEAREND in the usual way:

```
@HTTP://HOST.DOMAIN/YEAREND.SQL VAL1 VAL2
@FTP://HOST.DOMAIN/YEAREND.SQL  VAL1 VAL2
@GOPHER://HOST.DOMAIN/YEAREND.SQL VAL1 VAL2
```

On a web server configured to serve SQL reports, you could request SQL*Plus to execute a dynamic script by using:

```
@HTTP://HOST.DOMAIN/SCRIPTSERVER?ENDOFYEAR VAL1 VAL2
```

@@ (double “at” sign)

Syntax

@@file_name[*.ext*]

Runs a command file. This command is identical to the @ (“at” sign) command. It is useful for running nested command files because it has the additional functionality of looking for the specified command file in the same path or *uri* as the command file from which it was called. *uri* is only supported on Windows platforms in this release.

Terms

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 or from the same *uri* as the command file from which it was called. 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. See the Oracle installation and user’s manual provided for your operating system for specific information related to your operating system environment.

Usage

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.

Examples

Suppose that you have the following command file named PRINTRPT:

```
SELECT DEPARTMENT_ID, CITY FROM EMP_DETAILS_VIEW WHERE SALARY>12000;
@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.

Suppose that the same command file PRINTRPT was located on a web server and you ran it with START HTTP://HOST.DOMAIN/PRINTRPT. When it reaches the @ command, it looks for the command file named EMPRPT in the current local working directory and runs it. When PRINTRPT reaches the @@ command, it looks for the command file named WKRPT in the same *uri* as PRINTRPT and runs it.

/ (slash)

Syntax

/

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

Usage

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.

Examples

Type the following SQL query:

```
SELECT CITY, COUNTRY_NAME
FROM EMP_DETAILS_VIEW
WHERE SALARY=12000;
```

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

```
/
```

```
CITY                           COUNTRY_NAME
------------------------------ ----------------------------------------
Seattle                        United States of America
Oxford                         United Kingdom
Seattle                        United States of America
```

ACCEPT

Syntax

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

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

Terms

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 *Oracle9i Administrator's*

Guide and the *Oracle9i 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

```
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

```
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/yyyy" and a default of "01/01/2001", enter

```
ACCEPT hired DATE FORMAT 'dd/mm/yyyy' DEFAULT '01/01/2001'-
PROMPT 'Enter date hired:  '
```

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

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

APPEND

Syntax

A[PPEND] *text*

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

Terms

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

text

Represents the text to append. 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 CITY to the first line of the buffer, make that line the current line by listing the line as follows:

```
1
```

```
1* SELECT DEPARTMENT_ID
```

Now enter APPEND:

```
APPEND  CITY
1
```

```
1* SELECT DEPARTMENT_ID, CITY
```

Notice the double space between APPEND and CITY. 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

```
APPEND ;;
```

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

ARCHIVE LOG

Syntax

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

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

Terms

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:

```
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 redo 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 re-archive 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

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 *Oracle9i SQL Reference.*

Examples

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

```
ARCHIVE LOG START
```

To stop automatic archiving, enter

```
ARCHIVE LOG STOP
```

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

```
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

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

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.

Terms

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

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 LAST_NAME attribute of the Object Type EMPLOYEE_TYPE 20 characters wide, enter

```
ATTRIBUTE EMPLOYEE_TYPE.LAST_NAME FORMAT A20
```

To format the SALARY attribute of the Object Type EMPLOYEE_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

```
ATTRIBUTE EMPLOYEE_TYPE.SALARY FORMAT $9,999,990.99
```

BREAK

Syntax

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

where:

report_element	Requires the following syntax: {*column*|*expr*|ROW|REPORT}
action	Requires the following syntax: [SKI[P] *n*|[SKI[P]] PAGE][NODUP[LICATES]|DUP[LICATES]]

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.

Terms

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, or 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

```
BREAK ON DEPARTMENT_ID SKIP PAGE ON JOB_ID -
```

```
SKIP 1 ON SALARY SKIP 1
```

the first ON clause represents the *outermost break* (in this case, ON DEPARTMENT_ID) and the last ON clause represents the *innermost break* (in this case, ON SALARY). 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 DEPARTMENT_ID, then JOB_ID, then SALARY.

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 SALARY toward SKIP PAGE for ON DEPARTMENT_ID). 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_ID changes—but the values of DEPARTMENT_ID and SALARY remain the same—SQL*Plus skips *two* lines before printing the row (one as a result of SKIP 1 ON SALARY and one as a result of SKIP 1 ON JOB_ID).

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.

If the BREAK command specified earlier in this section is used, the following SELECT command produces meaningful results:

```
SELECT DEPARTMENT_ID, JOB_ID, SALARY, LAST_NAME
FROM EMP_DETAILS_VIEW
WHERE SALARY > 12000
ORDER BY DEPARTMENT_ID, JOB_ID, SALARY, LAST_NAME;
```

All rows with the same DEPARTMENT_ID print together on one page, and within that page all rows with the same JOB_ID print in groups. Within each group of jobs, those jobs with the same SALARY print in groups. Breaks in LAST_NAME cause no action because LAST_NAME 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[LICATES]

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

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

Usage

Each new BREAK command you enter replaces the preceding one.

To remove the BREAK command, use CLEAR BREAKS.

Examples

To produce a report that prints duplicate job values, prints the average of SALARY and inserts one blank line when the value of JOB_ID changes, and *additionally* prints the sum of SALARY and inserts another blank line when the value of DEPARTMENT_ID changes, you could enter the following commands. (The example selects departments 50 and 80 and the jobs of clerk and salesman only.)

```
BREAK ON DEPARTMENT_ID SKIP 1 ON JOB_ID SKIP 1 DUPLICATES
COMPUTE SUM OF SALARY ON DEPARTMENT_ID
COMPUTE AVG OF SALARY ON JOB_ID
SELECT DEPARTMENT_ID, JOB_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE JOB_ID IN ('SH_CLERK', 'SA_MAN')
AND DEPARTMENT_ID IN (50, 80)
ORDER BY DEPARTMENT_ID, JOB_ID;
```

```
DEPARTMENT_ID JOB_ID     LAST_NAME                    SALARY
------------- ---------- ------------------------ ----------
           50 SH_CLERK   Taylor                         3200
              SH_CLERK   Fleaur                        3100
                 .
                 .
                 .
              SH_CLERK   Gates                          2900

DEPARTMENT_ID JOB_ID     LAST_NAME                    SALARY
------------- ---------- ------------------------ ----------
           50 SH_CLERK   Perkins                        2500
              SH_CLERK   Bell                           4000
                 .
                 .
                 .
              SH_CLERK   Grant                          2600
              **********                             ----------
              avg                                         3215

DEPARTMENT_ID JOB_ID     LAST_NAME                    SALARY
------------- ---------- ------------------------ ----------

*************                                        ----------
sum                                                      64300

           80 SA_MAN     Russell                       14000
              SA_MAN     Partners                      13500
              SA_MAN     Errazuriz                     12000
              SA_MAN     Cambrault                     11000
              SA_MAN     Zlotkey                       10500
              **********                             ----------
              avg                                         12200

DEPARTMENT_ID JOB_ID     LAST_NAME                    SALARY
------------- ---------- ------------------------ ----------

*************                                        ----------
sum                                                      61000


25 rows selected.
```

BTITLE

Syntax

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

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, "Obsolete SQL*Plus Commands".

Terms

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

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

```
BTITLE LEFT 'CORPORATE PLANNING DEPARTMENT' -
RIGHT '1 JAN 2001'
```

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

```
BTITLE COL 50 'CONFIDENTIAL' TAB 6 '1 JAN 2001'
```

CHANGE

Syntax

C[HANGE] *sepchar old* [*sepchar* [*new* [*sepchar*]]]

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

Terms

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*.

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

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

Enter 3 so the current line of the buffer contains the following text:

```
3
```

```
  3* WHERE JOB_ID IS IN ('CLERK', 'SA_MAN')
```

Enter the following command:

```
CHANGE /CLERK/SH_CLERK/
```

The text in the buffer changes as follows:

```
  3*  WHERE JOB_ID IS IN ('SH_CLERK', 'SA_MAN')
```

Or enter the following command:

```
CHANGE /'CLERK',... /'SH_CLERK'/
```

The original line changes to

```
  3* WHERE JOB_ID IS IN ('SH_CLERK')
```

Or enter the following command:

```
CHANGE /(...)/('SA_MAN')/
```

The original line changes to

```
  3* WHERE JOB_ID IS IN ('SA_MAN')
```

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

```
3  WHERE JOB_ID IS IN ('SH_CLERK')
```

causes the second line of the buffer to be replaced with

```
WHERE JOB_ID IS IN ('SH_CLERK')
```

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

```
2  CHANGE/OLD/NEW/
```

will change the second line of the buffer to be

```
2* C/OLD/NEW/
```

CLEAR

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]

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

Terms

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, "Obsolete SQL*Plus Commands").

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, "Obsolete SQL*Plus Commands").

TIMI[NG]

Deletes all timers created by the TIMING command.

Examples

To clear breaks, enter

```
CLEAR BREAKS
```

To clear column definitions, enter

```
CLEAR COLUMNS
```

COLUMN

Syntax

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

where *option* represents one of the following clauses:

ALI[AS] *alias*
CLE[AR]
ENTMAP {ON|OFF}
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]

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.

Terms

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 LAST_NAME applies to all columns named LAST_NAME 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.

ENTMAP {ON|OFF}

Allows entity mapping to be turned on or off for selected columns in HTML output. This feature allows you to include, for example, HTML hyperlinks in a column of data, while still mapping entities in other columns of the same report. By turning entity mapping off for a column containing HTML hyperlinks, the HTML anchor tag delimiters, <, >, " and &, are correctly interpreted in the report. Otherwise they would be replaced with their respective entities, <, >, " and &, preventing web browsers from correctly interpreting the HTML.

Entities in the column heading and any COMPUTE labels or output appearing in the column are mapped or not mapped according to the value of ENTMAP for the column.

The default setting for COLUMN ENTMAP is the current setting of the MARKUP HTML ENTMAP option. For more information about the MARKUP HTML ENTMAP option, see MARKUP Options in Chapter 7 and SET later this Chapter.

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 A*n*. (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 Oracle9*i*, 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 Oracle9*i* 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 A*n*. 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	Examples	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.

Table 8–1 Number Formats

Element	Examples	Description
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.
. (period)	`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 *n* is number of “9”s after “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 LAST_NAME 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 val-

ues 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 FORMAT command.

NOPRI[NT]|PRI[NT]

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

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 LAST_NAME column 20 characters wide and display EMPLOYEE NAME on two lines as the column heading, enter

```
COLUMN LAST_NAME FORMAT A20 HEADING 'EMPLOYEE|NAME'
```

To format the SALARY 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, enter

```
COLUMN SALARY 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

```
COLUMN SALARY+COMMISSION_PCT+BONUS-EXPENSES-INS-TAX ALIAS NET
```

```
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 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

```
COLUMN REMARKS FORMAT A20 WRAP
```

```
CUSTOMER   DATE        QUANTITY REMARKS
---------- ---------   -------- --------------------
123        25-AUG-2001      144 This order must be s
                                hipped by air freigh
                                t to ORD
```

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

```
CUSTOMER   DATE        QUANTITY REMARKS
---------- ---------   -------- ---------------------
123        25-AUG-2001      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-2001      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. Use the EMPLOYEES table of the HR schema in this case instead of EMP_DETAILS_VIEW as you have used up to now. (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.)

```
COLUMN JOB_ID NOPRINT NEW_VALUE JOBVAR
COLUMN TODAY  NOPRINT NEW_VALUE DATEVAR
BREAK ON JOB_ID SKIP PAGE ON TODAY
TTITLE CENTER 'Job Report' RIGHT DATEVAR  SKIP 2 -
LEFT 'Job:     ' JOBVAR SKIP 2
SELECT TO_CHAR(SYSDATE, 'MM/DD/YYYY') TODAY,
LAST_NAME, JOB_ID, MANAGER_ID, HIRE_DATE, SALARY, DEPARTMENT_ID
```

```
FROM EMPLOYEES WHERE JOB_ID IN ('MK_MAN', 'SA_MAN')
ORDER BY JOB_ID, LAST_NAME;
```

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

```
                                    Job Report
04/19/01

Job:     MK_MAN

LAST
NAME                 MANAGER_ID HIRE_DATE         SALARY DEPARTMENT_ID
-------------------- ---------- --------- -------------- -------------
Hartstein                   100 17-FEB-96     $13,000.00            20
                                           --------------
                                              $13,000.00

                                  Job Report                          04/19/01

Job:     SA_MAN

LAST
NAME                 MANAGER_ID HIRE_DATE         SALARY DEPARTMENT_ID
-------------------- ---------- --------- -------------- -------------
Errazuriz                   100 10-MAR-97     $12,000.00            80
Zlotkey                     100 29-JAN-00     $10,500.00            80
Cambrault                   100 15-OCT-99     $11,000.00            80
Russell                     100 01-OCT-96     $14,000.00            80
Partners                    100 05-JAN-97     $13,500.00            80
                                           --------------

                                  Job Report                          04/19/01

Job:     SA_MAN

LAST
NAME                 MANAGER_ID HIRE_DATE         SALARY DEPARTMENT_ID
-------------------- ---------- --------- -------------- -------------
                                              $12,200.00

6 rows selected.
```

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

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

```
Session altered.
```

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

```
SELECT HIRE_DATE
FROM EMPLOYEES
WHERE EMPLOYEE_ID = 206;
```

```
                                        Job Report                              04/19/01

Job:     SA_MAN

HIRE_DATE
----------
1994-06-07
```

See the *Oracle9i 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

Syntax

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

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

Terms

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.

COMPUTE command functions are always executed in the sequence AVG, COUNT, MINIMUM, MAXIMUM, NUMBER, SUM, STD, VARIANCE, regardless of their order in the COMPUTE command.

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
MIN[IMUM]	Minimum value	NUMBER, CHAR, NCHAR, VARCHAR2 (VARCHAR), NVARCHAR2 (NCHAR VARYING)
MAX[IMUM]	Maximum value	NUMBER, CHAR, NCHAR, VARCHAR2 (VARCHAR), NVARCHAR2 (NCHAR VARYING)
NUM[BER]	Count of rows	all types

Table 8–2 COMPUTE Functions

Function	Computes	Applies to Datatypes
SUM	Sum of non-null values	NUMBER
STD	Standard deviation 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. You must place single quotes around *text* containing spaces or punctuation. The label prints left justified and truncates to the column width or linesize, whichever is smaller. The maximum label length 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

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 “account manager” and “salesman” job classifications with a compute label of “TOTAL”, enter

```
BREAK ON JOB_ID SKIP 1;
COMPUTE SUM LABEL 'TOTAL' OF SALARY ON JOB_ID;
SELECT JOB_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE JOB_ID IN ('AC_MGR', 'SA_MAN')
ORDER BY JOB_ID, SALARY;
```

```
JOB_ID     LAST_NAME                     SALARY
---------- ------------------------- ----------
AC_MGR     Higgins                        12000
**********                           ----------
TOTAL                                     12000

SA_MAN     Zlotkey                        10500
           Cambrault                      11000
```

```
          Errazuriz                      12000
          Partners                       13500
          Russell                        14000
**********                            ----------
TOTAL                                    61000

6 rows selected.
```

To calculate the total of salaries greater than 12,000 on a report, enter

```
COMPUTE SUM OF SALARY ON REPORT
BREAK ON REPORT
COLUMN DUMMY HEADING ''
SELECT '   ' DUMMY, SALARY, EMPLOYEE_ID
FROM EMP_DETAILS_VIEW
WHERE SALARY > 12000
ORDER BY SALARY;
```

```
        SALARY EMPLOYEE_ID
--- ---------- -----------
         13000         201
         13500         146
         14000         145
         17000         101
         17000         102
         24000         100
    ----------
sum      98500

6 rows selected.
```

To calculate the average and maximum salary for the executive and accounting departments, enter

```
BREAK ON DEPARTMENT_NAME SKIP 1
COMPUTE AVG LABEL 'Dept Average' -
        MAX LABEL 'Dept Maximum' -
        OF SALARY ON DEPARTMENT_NAME
SELECT DEPARTMENT_NAME, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE DEPARTMENT_NAME IN ('Executive', 'Accounting')
ORDER BY DEPARTMENT_NAME;
```

```
DEPARTMENT_NAME                LAST_NAME                     SALARY
------------------------------ ------------------------- ----------
Accounting                     Higgins                        12000
                               Gietz                           8300
******************************                           ----------
Dept Average                                                  10150
Dept Maximum                                                  12000

Executive                      King                           24000
                               Kochhar                        17000
                               De Haan                        17000
******************************                           ----------
Dept Average                                                19333.3333
Dept Maximum                                                  24000
```

To sum salaries for departments <= 20 without printing the compute label, enter

```
COLUMN DUMMY NOPRINT
COMPUTE SUM OF SALARY ON DUMMY
BREAK ON DUMMY SKIP 1
SELECT DEPARTMENT_ID DUMMY, DEPARTMENT_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE DEPARTMENT_ID <= 20
ORDER BY DEPARTMENT_ID;
```

```
DEPARTMENT_ID LAST_NAME                     SALARY
------------- ------------------------- ----------
           10 Whalen                          4400
                                         ----------
                                              4400

           20 Hartstein                      13000
           20 Fay                             6000
                                         ----------
                                             19000
```

To total the salary at the end of the report without printing the compute label, enter

```
COLUMN DUMMY NOPRINT
COMPUTE SUM OF SALARY ON DUMMY
BREAK ON DUMMY
SELECT NULL DUMMY, DEPARTMENT_ID, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE DEPARTMENT_ID <= 30
ORDER BY DEPARTMENT_ID;
```

```
DEPARTMENT_ID LAST_NAME                    SALARY
------------- ------------------------ ----------
           10 Whalen                         4400
           20 Hartstein                     13000
           20 Fay                            6000
           30 Raphaely                      11000
           30 Khoo                           3100
           30 Baida                          2900
           30 Tobias                         2800
           30 Himuro                         2600
           30 Colmenares                     2500
                                        ----------
                                            48300

9 rows selected.
```

CONNECT

Syntax

CONN[ECT] [{ *logon* | / } [AS {SYSOPER|SYSDBA}]]

where *logon* requires the following syntax:

username[/*password*][@*connect_identifier*]

Connects a given username to Oracle.

Terms

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 "/ (slash)" 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.

connect_identifier

Consists of an Oracle Net connect identifier. The exact syntax depends upon the Oracle Net communications protocol your Oracle installation uses. For more information, refer to the Oracle Net 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.

/ (slash)

Represents a default logon using operating system authentication. You cannot enter a *connect_identifier* 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 *Oracle9i 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. You can use either of these privileged connections with the default logon, /. For information about system privileges, see the *Oracle9i Administrator's Guide.*

Usage

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 *Oracle9i SQL Reference.*

Examples

To connect across Oracle Net with username HR and password HR to the database known by the Oracle Net alias as FLEETDB, enter

```
CONNECT HR/HR@FLEETDB
```

To connect as user HR, letting SQL*Plus prompt you for the password, enter

```
CONNECT HR
```

For more information about setting up your password file, refer to the *Oracle9i Administrator's Guide.*

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

```
CONNECT HR/HR AS SYSDBA
```

To connect to an instance on the current node as a privileged default user, enter

```
CONNECT / AS SYSDBA
```

In the last two examples, your default schema becomes SYS.

COPY

Syntax

COPY {FROM *database* | TO *database* | FROM *database* TO *database*}
{APPEND|CREATE|INSERT|REPLACE} *destination_table* [(*column*, *column*, *column*, ...)]
USING query

where *database* has the following syntax:

username[/password]@connect_identifier

Copies data from a query to a table in a local or remote database. COPY supports the following datatypes:

CHAR
DATE
LONG
NUMBER
VARCHAR2

The COPY command is not being enhanced to handle datatypes or features introduced with, or after Oracle8. The COPY command is likely to be made obsolete in a future release.

Terms

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

FROM *database*

Specifies the 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 *database*

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.

database

Specifies *username[/password] @connect_identifier* of the Oracle source or destination database you wish to COPY FROM or COPY TO. 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 password response.

You must include the *connect_identifier* clause which consists of an Oracle Net connection string, to specify the source or destination database. The exact syntax depends upon the Oracle Net communications protocol your Oracle installation uses. For more information, refer to the Oracle Net manual appropriate for your protocol or contact your DBA.

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.

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.

Usage

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

```
COPY FROM HR/HR@HQ TO JOHN/CHROME@WEST -
REPLACE WESTEMPLOYEES -
USING SELECT * FROM EMPLOYEES
```

The following command copies selected records from EMPLOYEES to the database to which SQL*Plus is connected. SQL*Plus creates SALESMEN through the copy.

SQL*Plus copies only the columns EMPLOYEE_ID and LAST_NAME, and at the destination names them EMPLOYEE_ID and SA_MAN.

```
COPY FROM HR/HR@ORACLE01 -
CREATE SALESMEN (EMPLOYEE_ID, SA_MAN) -
USING SELECT EMPLOYEE_ID, LAST_NAME FROM EMPLOYEES -
WHERE JOB_ID='SA_MAN';
```

DEFINE

Syntax

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

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

Terms

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

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

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

as

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

You should avoid defining variables with names that may be identical to values that you will pass to them, as unexpected results can occur. If a value supplied for a defined variable matches a variable name, then the contents of the matching variable are used instead of the supplied value.

Some variables are predefined when SQL*Plus starts. Enter DEFINE to see their definitions.

Examples

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

```
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 DEPARTMENT_ID, type:

```
DEFINE DEPARTMENT_ID = 20
```

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

To list the definition of DEPARTMENT_ID, enter

```
DEFINE DEPARTMENT_ID
DEFINE DEPARTMENT_ID = "20" (CHAR)
```

This result shows that the value of DEPARTMENT_ID is 20.

DEL

Syntax

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

Deletes one or more lines of the buffer.

Terms

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

n	Deletes line *n*.
n m	Deletes lines *n* through *m*.
n *	Deletes line *n* through the current line.
n LAST	Deletes line *n* through the last line.
*	Deletes the current line.
* *n*	Deletes the current line through line *n*.
* 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

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 *Oracle9i SQL Reference*.

Examples

Assume the SQL buffer contains the following query:

```
SELECT LAST_NAME, DEPARTMENT_ID
FROM EMP_DETAILS_VIEW
WHERE JOB_ID = 'SA_MAN'
ORDER BY DEPARTMENT_ID;
```

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

```
LIST 3
  3* WHERE JOB_ID = 'SA_MAN'
```

followed by

```
DEL
```

The SQL buffer now contains the following lines:

```
SELECT LAN_NAME, DEPARTMENT_ID
FROM EMP_DETAILS_VIEW
ORDER BY DEPARTMENT_ID
```

To delete the third line of the buffer, enter

```
DEL 3
```

The SQL buffer now contains the following lines:

```
SELECT LAST_NAME, DEPARTMENT_ID
FROM EMP_DETAILS_VIEW
```

DESCRIBE

Syntax

DESC[RIBE] {[*schema*.]*object*[*@connect_identifier*]}

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

Terms

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.

@connect_identifier

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 *Oracle9i SQL Reference*.

Usage

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 input or output, and default values, if any

Examples

To describe the view EMP_DETAILS_VIEW, enter

```
DESCRIBE EMP_DETAILS_VIEW
```

```
 Name                                    Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE_ID                             NOT NULL NUMBER(6)
 JOB_ID                                  NOT NULL VARCHAR2(10)
 MANAGER_ID                                       NUMBER(6)
 DEPARTMENT_ID                                    NUMBER(4)
 LOCATION_ID                                      NUMBER(4)
 COUNTRY_ID                                       CHAR(2)
 FIRST_NAME                                       VARCHAR2(20)
 LAST_NAME                               NOT NULL VARCHAR2(25)
 SALARY                                           NUMBER(8,2)
 COMMISSION_PCT                                   NUMBER(2,2)
 DEPARTMENT_NAME                         NOT NULL VARCHAR2(30)
 JOB_TITLE                               NOT NULL VARCHAR2(35)
 CITY                                    NOT NULL VARCHAR2(30)
 STATE_PROVINCE                                   VARCHAR2(25)
 COUNTRY_NAME                                     VARCHAR2(40)
 REGION_NAME                                      VARCHAR2(25)
```

To describe a procedure called CUSTOMER_LOOKUP, enter

```
DESCRIBE customer_lookup
```

```
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

```
CREATE PACKAGE apack AS
PROCEDURE aproc(P1 CHAR, P2 NUMBER);
PROCEDURE bproc(P1 CHAR, P2 NUMBER);
END apack;
/
```

```
Package created.
```

```
DESCRIBE apack
```

```
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

```
CREATE TYPE ADDRESS AS OBJECT
( STREET  VARCHAR2(20),
CITY    VARCHAR2(20)
);
/
```

```
Type created.
```

```
DESCRIBE address
```

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

To create and describe the object type EMPLOYEE that contains the attributes LAST_NAME, EMPADDR, JOB_ID and SALARY, enter

```
CREATE TYPE EMPLOYEE AS OBJECT
(LAST_NAME VARCHAR2(30),
EMPADDR ADDRESS,
JOB_ID VARCHAR2(20),
SALARY NUMBER(7,2)
);
/
```

```
Type created.
```

```
DESCRIBE employee
```

```
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LAST_NAME                                          VARCHAR2(30)
 EMPADDR                                            ADDRESS
 JOB_ID                                             VARCHAR2(20)
 SALARY                                             NUMBER(7,2)
```

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

```
CREATE TYPE addr_type IS TABLE OF ADDRESS;
/
```

```
Type created.
```

```
DESCRIBE addr_type
```

```
 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

```
CREATE TYPE addr_varray AS VARRAY(10) OF ADDRESS;
/
```

```
Type created.
```

```
DESCRIBE addr_varray
```

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

To create and describe the table department that contains the columns DEPARTMENT_ID, PERSON and LOC, enter

```
CREATE TABLE department
(DEPARTMENT_ID NUMBER,
PERSON EMPLOYEE,
LOC NUMBER
);
/
```

```
Table created.
```

```
DESCRIBE department
```

```
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DEPARTMENT_ID                                      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

```
CREATE OR REPLACE TYPE rational AS OBJECT
(NUMERATOR NUMBER,
DENOMINATOR NUMBER,
MAP MEMBER FUNCTION rational_order -
RETURN DOUBLE PRECISION,
PRAGMA RESTRICT_REFERENCES
(rational_order, RNDS, WNDS, RNPS, WNPS) );
/
CREATE OR REPLACE TYPE BODY rational AS OBJECT
MAP MEMBER FUNCTION rational_order -
RETURN DOUBLE PRECISION IS
BEGIN
  RETURN NUMERATOR/DENOMINATOR;
END;
END;
/
```

```
DESCRIBE rational
```

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

METHOD
------
MAP MEMBER FUNCTION RATIONAL_ORDER RETURNS NUMBER
```

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

```
SET LINESIZE 80
SET DESCRIBE DEPTH 2
SET DESCRIBE INDENT ON
SET DESCRIBE LINE OFF
```

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

```
SHOW DESCRIBE
```

```
describe DEPTH  2 LINENUM OFF INDENT ON
```

```
DESCRIBE employee
```

```
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FIRST_NAME                                         VARCHAR2(30)
 EMPADDR                                            ADDRESS
   STREET                                           VARCHAR2(20)
   CITY                                             VARCHAR2(20)
 JOB_ID                                             VARCHAR2(20)
 SALARY                                             NUMBER(7,2)
```

For more information on using the CREATE TYPE command, see your *Oracle9i SQL Reference.*

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

DISCONNECT

Syntax

DISC[ONNECT]

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

Usage

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.

Examples

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

```
CONNECT HR/HR
SELECT LAST_NAME, DEPARTMENT_NAME FROM EMP_DETAILS_VIEW;
DISCONNECT
SET INSTANCE FIN2
CONNECT HR2/HR2
```

EDIT

Syntax

ED[IT] [*file_name*[.*ext*]]

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

Terms

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

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.

Examples

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

```
EDIT REPORT
```

EXECUTE

Syntax

EXEC[UTE] *statement*

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*.

Terms

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

statement

Represents a PL/SQL statement.

Usage

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

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

Examples

If the variable :n has been defined with:

```
VARIABLE n NUMBER
```

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

```
EXECUTE :n := 1
```

```
PL/SQL procedure successfully completed.
```

For information on how to create a bind variable, see the VARIABLE command in this chapter.

EXIT

Syntax

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

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

Terms

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

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.

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.

Examples

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

```
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

Syntax

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

Loads a host operating system file into the SQL buffer.

Terms

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 after it is loaded. This is the default.

NOL[IST]

Suppresses the listing.

Usage

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 the filename you specify contains the word *list* or the word *file*, the name must be 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.

Examples

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

```
GET YEARENDRPT
```

HELP

Syntax

HELP [*topic*]

Accesses the SQL*Plus help system. Enter HELP INDEX for a list of topics.

Terms

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

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

```
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.

Examples

To see a list of SQL*Plus commands for which help is available, enter

```
HELP INDEX
```

Alternatively, to see a single column display of SQL*Plus commands for which help is available, enter

```
HELP TOPICS
```

HOST

Syntax

HO[ST] [*command*]

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

Terms

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.

Note: Operating system commands entered from a SQL*Plus session using the HOST command do not effect the current SQL*Plus session. For example, setting an operating system environment variable does not effect the current SQL*Plus session, it only effects SQL*Plus sessions started subsequently.

You can suppress access to the HOST command. For more information about suppressing the HOST command see Appendix E, "Security".

Usage

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.

Examples

To execute a UNIX operating system command, ls *.sql, enter

```
HOST ls *.sql
```

To execute a Windows operating system command, dir *.sql, enter

```
HOST dir *.sql
```

INPUT

Syntax

I[NPUT] [*text*]

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

Terms

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 or a period.

Usage

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:

```
SELECT LAST_NAME, DEPARTMENT_ID, SALARY, COMMISSION_PCT
FROM EMP_DETAILS_VIEW
```

To add an ORDER BY clause to the query, enter

```
LIST 2
```

```
  2* FROM EMP_DETAILS_VIEW
```

```
INPUT ORDER BY LAST_NAME
```

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 LAST_NAME, DEPARTMENT_ID, SALARY, COMMISSION_PCT
2  FROM  EMP_DETAILS_VIEW
3* ORDER BY LAST_NAME
```

To add a two-line WHERE clause, enter

```
LIST 2
```

```
  2* FROM EMP_DETAILS_VIEW
```

```
INPUT
  3  WHERE JOB_ID = 'SA_MAN'
  4  AND COMMISSION_PCT=.25
  5
```

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

```
SELECT LAST_NAME, DEPARTMENT_ID, SALARY, COMMISSION_PCT
FROM EMP_DETAILS_VIEW
WHERE JOB_ID = 'SA_MAN'
AND COMMISSION_PCT = .25
ORDER BY LAST_NAME
```

LIST

Syntax

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

Lists one or more lines of the SQL buffer.

Terms

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

n	Lists line *n*.
n m	Lists lines *n* through *m*.
n *	Lists line *n* through the current line.
n LAST	Lists line *n* through the last line.
*	Lists the current line.
* *n*	Lists the current line through line *n*.
* LAST	Lists the current line through the last line.
LAST	Lists the last line.

Enter LIST with no clauses to list all lines. The last line or only line listed becomes the new current line (marked by an asterisk).

Examples

To list the contents of the buffer, enter

```
LIST
```

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

```
  1  SELECT LAST_NAME, DEPARTMENT_ID, JOB_ID
  2  FROM EMP_DETAILS_VIEW
  3  WHERE JOB_ID = 'SH_CLERK'
  4* ORDER BY DEPARTMENT_ID
```

The asterisk indicates that line 4 is the current line.

To list the second line only, enter

```
LIST 2
```

The second line is displayed:

```
2* FROM EMP_DETAILS_VIEW
```

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

```
LIST * LAST
```

You will then see this:

```
2  FROM EMP_DETAILS_VIEW
3  WHERE JOB_ID = 'SH_CLERK'
4* ORDER BY DEPARTMENT_ID
```

PASSWORD

Syntax

PASSW[ORD] [*username*]

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

Terms

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

username

Specifies the user. If omitted, *username* defaults to the current user.

Usage

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.

Examples

Suppose you are logged on as HR/HR, and want to change the password to HRHR

```
PASSWORD
Changing password for HR
Old password: HR
New password: HRHR
Retype new password: HRHR
Password changed
```

Suppose you are logged on as a DBA, and want to change the password for user johnw (currently identified by johnwpass) to johnwnewpass

```
PASSWORD johnw
Changing password for johnw
New password: johnwnewpass
Retype new password: johnwnewpass
Password changed
```

Note: Passwords are not echoed to the screen, they are shown here for your convenience.

PAUSE

Syntax

PAU[SE] [*text*]

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.

Terms

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

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.

Examples

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

Syntax

PRI[NT] [*variable* ...]

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

Terms

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

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.

Examples

The following example illustrates a PRINT command:

```
VARIABLE n NUMBER
BEGIN
:n := 1;
END;
/
```

```
PL/SQL procedure successfully completed.
```

```
PRINT n
         N
----------
         1
```

PROMPT

Syntax

PRO[MPT] [*text*]

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

Terms

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

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

Examples

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 'Department ID?> '
SELECT DEPARTMENT_NAME FROM EMP_DETAILS_VIEW
WHERE DEPARTMENT_ID = &NEWDEPT
```

Assume you run the file using START or @:

```
@ASKFORDEPT
```

```
Please enter a valid department
For example:  10, 20, 30, 40
Department ID?>
```

You can enter a department number at the prompt Department ID?>. 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 Department ID?> prompt. You can use SET VERIFY OFF to prevent this behavior.

RECOVER

Syntax

RECOVER {*general* | *managed* | END BACKUP}

where the *general* clause has the following syntax:

[AUTOMATIC] [FROM *location*]
{ {*full_database_recovery* | *partial_database_recovery* |LOGFILE *filename*}
[{TEST | ALLOW integer CORRUPTION } [TEST | ALLOW integer CORRUPTION]...]
|CONTINUE [DEFAULT]|CANCEL}

where the *full_database_recovery* clause has the following syntax:

[STANDBY] DATABASE
[{UNTIL {CANCEL | TIME *date* | CHANGE *integer*} | USING BACKUP CONTROLFILE}
[UNTIL {CANCEL | TIME *date* | CHANGE *integer*} | USING BACKUP CONTROLFILE]...]

where the *partial_database_recovery* clause has the following syntax:

{TABLESPACE *tablespace* [, *tablespace*]... | DATAFILE *datafilename* [, *datafilename*]...
| STANDBY
{TABLESPACE *tablespace* [, *tablespace*]... | DATAFILE *datafilename* [, *datafilename*]...}
UNTIL [CONSISTENT] [WITH] CONTROLFILE }

where the *managed* clause has the following syntax:

MANAGED STANDBY DATABASE
[{NODELAY | [TIMEOUT] *integer* | CANCEL [IMMEDIATE] [NOWAIT]}
| [DISCONNECT [FROM SESSION]] [FINISH [NOWAIT]]]

Performs media recovery on one or more tablespaces, one or more datafiles, or the entire database. For more information on the RECOVER command, see the *Oracle9i Administrator's Guide*, the ALTER DATABASE RECOVER command in the *Oracle9i SQL Reference*, the *Oracle9i Backup and Recovery Concepts* guide, and the *Oracle9i User-Managed Backup and Recovery Guide*.

Terms

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, SQL*Plus prompts you for a filename, displaying a generated filename as a suggestion.

If you do not specify either AUTOMATIC or LOGFILE, SQL*Plus prompts you for a filename, suggesting the generated filename. 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, SQL*Plus assumes the archived redo log file group is in the location specified by the initialization parameter LOG_ARCHIVE_DEST or LOG_ARCHIVE_DEST_1. Do not specify FROM if you have set a file with SET LOGSOURCE.

LOGFILE

Continues media recovery by applying the specified redo log file. In interactive recovery mode (AUTORECOVERY OFF), if a bad log name is entered, errors for the bad log name are displayed and you are prompted to enter a new log name.

TEST ALLOW integer CORRUPTION

In the event of logfile corruption, specifies the number of corrupt blocks that can be tolerated while allowing recovery to proceed. During normal recovery, integer cannot exceed 1.

CONTINUE

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

CONTINUE DEFAULT

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

CANCEL

Terminates cancel-based recovery.

STANDBY DATABASE

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 suggested filenames of archived redo log files, and recovery completes when you specify CANCEL instead of a filename.

UNTIL TIME

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

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

UNTIL CHANGE

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.

USING BACKUP CONTROLFILE

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

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

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

STANDBY TABLESPACE

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

STANDBY DATAFILE

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

UNTIL CONSISTENT WITH CONTROLFILE

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

MANAGED STANDBY DATABASE

Specifies sustained standby recovery mode. This mode assumes that the standby database is an active component of an overall standby database architecture. A primary database actively archives its redo log files to the standby site. As these archived redo logs arrive at the standby site, they become available for use by a managed standby recovery operation. Sustained standby recovery is restricted to media recovery.

For more information on the parameters of this clause, see the *Oracle9i User-Managed Backup and Recovery Guide*.

NODELAY

Applies a delayed archivelog immediately to the standby database overriding any DELAY setting in the LOG_ARCHIVE_DEST_n parameter on the primary database. If you omit this clause, application of the archivelog is delayed according to the parameter setting. If DELAY was not specified in the parameter, the archivelog is applied immediately.

TIMEOUT

Specifies in minutes the wait period of the sustained recovery operation. The recovery process waits for integer minutes for a requested archived log redo to be available for writing to the standby database. If the redo log file does not become available within that time, the recovery process terminates with an error message. You can then issue the statement again to return to sustained standby recovery mode.

If you do not specify this clause, the database remains in sustained standby recovery mode until you reissue the statement with the RECOVER CANCEL clause or until instance shutdown or failure.

CANCEL (managed clause)

In managed recovery, CANCEL terminates the managed standby recovery operation after applying the current archived redo file. Session control returns when the recovery process terminates.

CANCEL IMMEDIATE

Terminates the managed recovery operation after applying the current archived redo file or after the next redo log file read, whichever comes first. Session control returns when the recovery process terminates.

CANCEL IMMEDIATE cannot be issued from the same session that issued the RECOVER MANAGED STANDBY DATABASE statement.

CANCEL NOWAIT

Terminates the managed recovery operation after the next redo log file read and returns session control immediately.

DISCONNECT FROM SESSION

Indicates that the managed redo process (MRP) should apply archived redo files as a detached background process. Doing so leaves the current session available.

FINISH

Recovers the current log standby logfiles of the standby database. It is useful in the event of the primary database failure. This clause overrides any delays specified for archivelogs, so that Oracle logs are applied immediately.

NOWAIT

Returns control immediately rather than after completion of the recovery process.

Usage

You must have the OSDBA role enabled. 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 and closed, and all tablespaces requiring recovery must be online.

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.

Examples

To recover the entire database, enter

```
RECOVER DATABASE
```

To recover the database until a specified time, enter

```
RECOVER DATABASE UNTIL TIME 01-JAN-2001:04:32:00
```

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

```
RECOVER TABLESPACE ts_one, ts_two
```

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

```
RECOVER DATAFILE 'data1.db'
```

REMARK

Syntax

REM[ARK]

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

Usage

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.

Examples

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_ID" ON REPORT
REM Each column displays the sums of salaries by job for
REM one of the departments 10, 20, 30.
SELECT JOB_ID,
SUM(DECODE( DEPARTMENT_ID, 10, SALARY, 0)) "DEPARTMENT 10",
SUM(DECODE( DEPARTMENT_ID, 20, SALARY, 0)) "DEPARTMENT 20",
SUM(DECODE( DEPARTMENT_ID, 30, SALARY, 0)) "DEPARTMENT 30",
SUM(SALARY) "TOTAL BY JOB_ID"
FROM EMP_DETAILS_VIEW
GROUP BY JOB_ID;
```

REPFOOTER

Syntax

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

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

Terms

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

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.

Examples

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

```
REPFOOTER PAGE CENTER 'END EMPLOYEE LISTING REPORT'
TTITLE RIGHT 'Page: ' FORMAT 999 SQL.PNO
SELECT LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY > 12000;
```

```
LAST_NAME                      SALARY
------------------------- ----------
King                            24000
Kochhar                         17000
De Haan                         17000
Russell                         14000
```

```
Partners                        13500
Hartstein                       13000
                           ----------
sum                             98500

                                                                Page:    2
                           END EMPLOYEE LISTING REPORT

6 rows selected.
```

To suppress the report footer without changing its definition, enter

```
REPFOOTER OFF
```

You can stop the REPHEADER display by setting it off with:

```
REPHEADER OFF
```

REPHEADER

Syntax

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

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*

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

Terms

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 is the current line number, SQL.PNO is the current page number, SQL.RELEASE is the current Oracle release number, SQL.CODE is the current error code, and SQL.USER is the current user-name.

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 bold 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 using 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

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, positioning and formatting each constant or variable as specified by the *printspec* clauses that precede it.

Examples

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

```
REPHEADER PAGE CENTER 'EMPLOYEE LISTING REPORT'
TTITLE RIGHT 'Page: ' FORMAT 999 SQL.PNO
SELECT LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY > 12000;
```

```
                                                              Page:    1
                            EMPLOYEE LISTING REPORT

                                                              Page:    2
LAST_NAME                      SALARY
-------------------------- ----------
King                            24000
Kochhar                         17000
De Haan                         17000
Russell                         14000
Partners                        13500
Hartstein                       13000
                           ----------
sum                             98500

6 rows selected.
```

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

```
REPHEADER OFF
```

RUN

Syntax

R[UN]

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

Usage

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.

Examples

Assume the SQL buffer contains the following query:

```
SELECT DEPARTMENT_ID
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000
```

To RUN the query, enter

```
RUN
```

```
  1  SELECT DEPARTMENT_ID
  2  FROM EMP_DETAILS_VIEW
  3* WHERE SALARY>12000

DEPARTMENT_ID
-------------
           90
           90
           90
           80
           80
           20

6 rows selected.
```

SAVE

Syntax

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

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

Terms

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.

CREATE

Creates a new file with the name specified. This is the default behavior.

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

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

```
SAVE DEPTSALRPT
```

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

```
SAVE DEPTSALRPT.OLD
```

SET

Syntax

SET *system_variable value*

where *system_variable* and *value* represent one of the following clauses.

APPI[NFO]{ON|OFF|text}
ARRAY[SIZE] {15|n}
AUTO[COMMIT]{ON|OFF|IMM[EDIATE]|n}
AUTOP[RINT] {ON|OFF}
AUTORECOVERY [ON|OFF]
AUTOT[RACE] {ON|OFF|TRACE[ONLY]} [EXP[LAIN]] [STAT[ISTICS]]
BLO[CKTERMINATOR] {.|c}
CMDS[EP] {;|c|ON|OFF}
COLSEP {_|text}
COM[PATIBILITY]{V7|V8|NATIVE}
CON[CAT] {.|c|ON|OFF}
COPYC[OMMIT] {0|n}
COPYTYPECHECK {ON|OFF}
DEF[INE] {&|c|ON|OFF}
DESCRIBE [DEPTH {1|n|ALL}][LINENUM {ON|OFF}][INDENT {ON|OFF}]
ECHO {ON|OFF}
EDITF[ILE] file_name[.ext]
EMB[EDDED] {ON|OFF}
ESC[APE] {\|c|ON|OFF}
FEED[BACK] {6|n|ON|OFF}
FLAGGER {OFF|ENTRY |INTERMED[IATE]|FULL}
FLU[SH] {ON|OFF}
HEA[DING] {ON|OFF}
HEADS[EP] {||c|ON|OFF}
INSTANCE [instance_path|LOCAL]
LIN[ESIZE] {80|n}
LOBOF[FSET] {n|1}
LOGSOURCE [pathname]
LONG {80|n}
LONGC[HUNKSIZE] {80|n}
MARK[UP] HTML [ON|OFF] [HEAD text] [BODY text] [TABLE text] [ENTMAP {ON|OFF}] [SPOOL {ON|OFF}] [PRE[FORMAT] {ON|OFF}]
NEWP[AGE] {1|n|NONE}
NULL text

```
NUMF[ORMAT] format
NUM[WIDTH] {10|n}
PAGES[IZE] {24|n}
PAU[SE] {ON|OFF|text}
RECSEP {WR[APPED]|EA[CH]|OFF}
RECSEPCHAR {_|c}
SERVEROUT[PUT] {ON|OFF} [SIZE n] [FOR[MAT] {WRA[PPED]|WOR[D_
WRAPPED]|TRU[NCATED]}]
SERVEROUT[PUT] {ON|OFF} [SIZE n] [FOR[MAT] {WRA[PPED]|WOR[D_
WRAPPED]|TRU[NCATED]}]
SHIFT[INOUT] {VIS[IBLE]|INV[ISIBLE]}
SHOW[MODE] {ON|OFF}
SQLBL[ANKLINES] {ON|OFF}
SQLC[ASE] {MIX[ED]|LO[WER]|UP[PER]}
SQLCO[NTINUE] {> |text}
SQLN[UMBER] {ON|OFF}
SQLPLUSCOMPAT[IBILITY] {x.y[.z]}
SQLPRE[FIX] {#|c}
SQLP[ROMPT] {SQL>|text}
SQLT[ERMINATOR] {;|c|ON|OFF}
SUF[FIX] {SQL|text}
TAB {ON|OFF}
TERM[OUT] {ON|OFF}
TI[ME] {ON|OFF}
TIMI[NG] {ON|OFF}
TRIM[OUT] {ON|OFF}
TRIMS[POOL] {ON|OFF}
UND[ERLINE] {-|c|ON|OFF}
VER[IFY] {ON|OFF}
WRA[P] {ON|OFF}
```

Sets a system variable to alter the SQL*Plus environment settings for your current session, for example:

- the display width for NUMBER data
- turning on HTML formatting
- enabling or disabling the printing of column headings
- the number of lines per page

Terms

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, which is the default. 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 Oracle9*i* database server product.

For more information on the DBMS_APPLICATION_INFO package, see the *Oracle9i Performance Guide and Reference* 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 lit-

tle added performance. ARRAYSIZE has no effect on the results of SQL*Plus operations other than increasing efficiency.

AUTO[COMMIT]{ON|OFF|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, or 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] {ON|OFF}

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. See the "RECOVER" command for more information about database recovery.

AUTOT[RACE] {ON|OFF|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. Information about EXPLAIN PLAN is documented in the *Oracle9i SQL Reference* manual.

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 *Oracle9i Performance Guide and Reference* 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*. It cannot be an alphanumeric character or a whitespace. To execute the block, you must issue a RUN or / (slash) command.

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

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 to V8 for Oracle8 or later. Set COMPATIBILITY to NATIVE if you wish the database to determine the setting (If connected to Oracle9*i*, compatibility defaults to NATIVE). COMPATIBILITY should be correctly set for the version of Oracle to which you are connected; otherwise, you may be unable to run any SQL commands.

Note: You can set COMPATIBILITY to V7 or V8 when connected to Oracle9*i*. This enables you to run Oracle7 SQL, Oracle8 or Oracle8*i* SQL against Oracle9*i*.

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

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 {ON|OFF}

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*|ON|OFF}

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, "Obsolete SQL*Plus Commands".

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 DESCRIBE earlier in this chapter.

ECHO {ON|OFF}

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] {ON|OFF}

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*|ON|OFF}

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*|ON|OFF}

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] {ON|OFF}

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] {ON|OFF}

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*|ON|OFF}

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 Oracle Net is running. You can use any valid Oracle Net connect identifier as the specified instance path. See your operating system-specific Oracle documentation for a complete description of how your operating system specifies Oracle Net connect identifiers. 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 in the Oracle initialization file, *init.ora*. 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.

MARK[UP] HTML [ON|OFF] [HEAD *text*] [BODY *text*] [TABLE *text*] [ENTMAP {ON|OFF}] [SPOOL {ON|OFF}] [PRE[FORMAT] {ON|OFF}]

Outputs HTML marked up text. SET MARKUP only specifies that SQL*Plus output will be HTML encoded. You must use SET MARKUP HTML ON SPOOL ON and the SQL*Plus SPOOL command to create and name a spool file, and to begin writing HMTL output to it. SET MARKUP has the same options and behavior as SQLPLUS -MARKUP.

For detailed information see "MARKUP Options" in Chapter 7. For examples of usage, see MARKUP on page 8-117, and "Creating Web Reports" in Chapter 4.

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.

PAU[SE] {ON|OFF|*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.

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 {WR[APPED]|EA[CH]|OFF}

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] {ON|OFF} [SIZE *n*] [FOR[MAT] {WRA[PPED]|WOR[D_WRAPPED]|TRU[NCATED]}]

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 Oracle8*i* or Oracle9*i* 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 bro-

ken 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 *Oracle9i Application Developer's Guide - Fundamentals*.

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

Allows correct alignment for terminals that display shift characters. The SET SHIFTINOUT 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] {ON|OFF}

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 or script. ON interprets blank lines and new lines as part of a SQL command or script. OFF, the default value, does not allow blank lines or new lines in a SQL command or script or script.

Enter the BLOCKTERMINATOR to stop SQL command entry without running the SQL command. Enter the SQLTERMINATOR character to stop SQL command entry and run the SQL statement.

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, to uppercase if SQLCASE equals UPPER,

to lowercase if SQLCASE equals LOWER, and makes no changes 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] {ON|OFF}

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.

SQLPLUSCOMPAT[IBILITY] {x.y[.z]}

Sets the behavior or output format of VARIABLE to that of the release or version specified by x.y[.z]. Where *x* is the *version* number, *y* is the *release* number, and *z* is the *update* number. For example, 8.1, 8.1.7 or 9.0.0. In later releases, SQLPLUSCOMPATIBILITY may affect features other than VARIABLE.

Setting the value of SQLPLUSCOMPATIBILITY to a version less than 9.0.0 will result in VARIABLE definition of NCHAR or NVARCHAR2 datatypes to revert to Oracle8*i* behavior whereby the size of the variable is in bytes or characters depending on the chosen national character set.

The default glogin.sql file contains SET SQLPLUSCOMPAT 8.1.7. It is recommended that you add SET SQLPLUSCOMPAT 9.0.0 to your scripts to maximize their compatibility with future versions of SQL*Plus.

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*|ON|OFF}

Sets the character used to end and execute SQL commands to *c*. It cannot be an alphanumeric character or a whitespace. OFF means that SQL*Plus recognizes no command terminator; you terminate a SQL command by entering an empty line. If SQLBLANKLINES is set ON, you must use the BLOCKTERMINATOR to terminate a SQL command. 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 {ON|OFF}

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] {ON|OFF}

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] {ON|OFF}

Controls the display of the current time. ON displays the current time before each command prompt. OFF suppresses the time display.

TIMI[NG] {ON|OFF}

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] {ON|OFF}

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] {ON|OFF}

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] {ON|OFF}

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

SQL*Plus maintains system variables (also called SET command variables) to enable you to setup 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 *Oracle9i 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, as it is SET OFF by default, enter

```
SET APPINFO ON
SHOW APPINFO
```

```
APPINFO is ON and set to "SQL*Plus"
```

To change the default text, enter

```
SET APPINFO 'This is SQL*Plus'
```

To make sure that registration has taken place, enter

```
VARIABLE MOD VARCHAR2(50)
VARIABLE ACT VARCHAR2(40)
EXECUTE DBMS_APPLICATION_INFO.READ_MODULE(:MOD, :ACT);
```

```
PL/SQL procedure successfully completed.
```

```
PRINT MOD
```

```
MOD
--------------------------------------------------
This is SQL*Plus
```

To change APPINFO back to its default setting, enter

```
SET APPI OFF
```

AUTORECOVERY

To set the recovery mode to AUTOMATIC, enter

```
SET AUTORECOVERY ON
RECOVER DATABASE
```

CMDSEP

To specify a title with TTITLE and format a column with COLUMN, both on the same line, enter

```
SET CMDSEP +
TTITLE LEFT 'SALARIES' + COLUMN SALARY FORMAT $99,999
```

```
SELECT LAST_NAME, SALARY FROM EMP_DETAILS_VIEW
WHERE JOB_ID = 'SH_CLERK';
```

```
SALARIES
LAST_NAME                 SALARY
------------------------- --------
Taylor                    $3,200
Fleaur                    $3,100
Sullivan                  $2,500
Geoni                     $2,800
Sarchand                  $4,200
Bull                      $4,100
Dellinger                 $3,400
Cabrio                    $3,000
Chung                     $3,800
Dilly                     $3,600
Gates                     $2,900
Perkins                   $2,500
Bell                      $4,000
Everett                   $3,900
McCain                    $3,200
Jones                     $2,800

SALARIES
LAST_NAME                 SALARY
------------------------- --------
Walsh                     $3,100
Feeney                    $3,000
OConnell                  $2,600
Grant                     $2,600

20 rows selected.
```

COLSEP

To set the column separator to "|" enter

```
SET COLSEP '|'
SELECT LAST_NAME, JOB_ID, DEPARTMENT_ID
FROM EMP_DETAILS_VIEW
WHERE DEPARTMENT_ID = 20;
```

```
LAST_NAME                |JOB_ID    |DEPARTMENT_ID
-------------------------|----------|-------------
Hartstein                |MK_MAN    |           20
Fay                      |MK_REP    |           20
```

COMPATIBILITY

To run a command file, SALARY.SQL, created with Oracle7 SQL syntax, enter

```
SET COMPATIBILITY V7
START SALARY
```

After running the file, reset compatibility to NATIVE to run command files created for Oracle9*i*:

```
SET COMPATIBILITY NATIVE
```

Alternatively, you can add the command SET COMPATIBILITY V7 to the beginning of the command file, and reset COMPATIBILITY to NATIVE at the end of the file.

DESCRIBE

To describe the view EMP_DETAILS_VIEW to a depth of two levels, and indent the output while also displaying line numbers, first describe the view as follows:

```
DESCRIBE EMP_DETAILS_VIEW
```

```
 Name                                    Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPLOYEE_ID                             NOT NULL NUMBER(6)
 JOB_ID                                  NOT NULL VARCHAR2(10)
 MANAGER_ID                                       NUMBER(6)
 DEPARTMENT_ID                                    NUMBER(4)
 LOCATION_ID                                      NUMBER(4)
 COUNTRY_ID                                       CHAR(2)
 FIRST_NAME                                       VARCHAR2(20)
 LAST_NAME                               NOT NULL VARCHAR2(25)
 SALARY                                           NUMBER(8,2)
 COMMISSION_PCT                                   NUMBER(2,2)
 DEPARTMENT_NAME                         NOT NULL VARCHAR2(30)
 JOB_TITLE                               NOT NULL VARCHAR2(35)
 CITY                                    NOT NULL VARCHAR2(30)
 STATE_PROVINCE                                   VARCHAR2(25)
 COUNTRY_NAME                                     VARCHAR2(40)
 REGION_NAME                                      VARCHAR2(25)
```

To format EMP_DETAILS_VIEW so that the output displays with indentation and line numbers, use the SET DESCRIBE command as follows:

```
SET DESCRIBE DEPTH 2 LINENUM ON INDENT ON
```

To display the above settings, enter

```
DESCRIBE EMP_DETAILS_VIEW
```

```
           Name                 Null?    Type
           -------------------- -------- --------------------
     1     EMPLOYEE_ID          NOT NULL NUMBER(6)
     2     JOB_ID               NOT NULL VARCHAR2(10)
     3     MANAGER_ID                    NUMBER(6)
     4     DEPARTMENT_ID                 NUMBER(4)
     5     LOCATION_ID                   NUMBER(4)
     6     COUNTRY_ID                    CHAR(2)
     7     FIRST_NAME                    VARCHAR2(20)
     8     LAST_NAME            NOT NULL VARCHAR2(25)
     9     SALARY                        NUMBER(8,2)
    10     COMMISSION_PCT                NUMBER(2,2)
    11     DEPARTMENT_NAME      NOT NULL VARCHAR2(30)
    12     JOB_TITLE            NOT NULL VARCHAR2(35)
    13     CITY                 NOT NULL VARCHAR2(30)
    14     STATE_PROVINCE                VARCHAR2(25)
    15     COUNTRY_NAME                  VARCHAR2(40)
    16     REGION_NAME                   VARCHAR2(25)
```

ESCAPE

If you define the escape character as an exclamation point (!), then

```
SET ESCAPE !
ACCEPT v1 PROMPT 'Enter !&1:'
```

displays this prompt:

```
Enter &1:
```

To set the escape character back to the default value of \ (backslash), enter

```
SET ESCAPE ON
```

HEADING

To suppress the display of column headings in a report, enter

```
SET HEADING OFF
```

If you then run a SQL SELECT command

```
SELECT LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE JOB_ID = 'AC_MGR';
```

the following output results:

```
Higgins                   12000
```

To turn the display of column headings back on, enter

```
SET HEADING ON
```

INSTANCE

To set the default instance to "PROD1" enter

```
DISCONNECT
SET INSTANCE PROD1
```

To set the instance back to the default of local, enter

```
SET INSTANCE local
```

You must disconnect from any connected instances to change the instance.

LOBOFFSET

To set the starting position from which a CLOB column's data is retrieved to the 22nd position, enter

```
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/oracle90/dbs/arch" enter

```
SET LOGSOURCE "/usr/oracle90/dbs/arch"
RECOVER DATABASE
```

LONG

To set the maximum number of characters to fetch for displaying and copying LONG values, to 500, enter

```
SET LONG 500
```

The LONG data will wrap on your screen; SQL*Plus will not truncate until the 501st character. The default for LONG is 80 characters.

LONGCHUNKSIZE

To set the size of the increments in which SQL*Plus retrieves LONG values to 100 characters, enter

```
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, whichever is the smaller.

MARKUP

The following is a script which uses the SET MARKUP HTML command to enable HTML marked up text to be spooled to a specified file:

> **Note:** The SET MARKUP example command is laid out for readability using line continuation characters "-" and spacing. Command options are concatenated in normal entry.

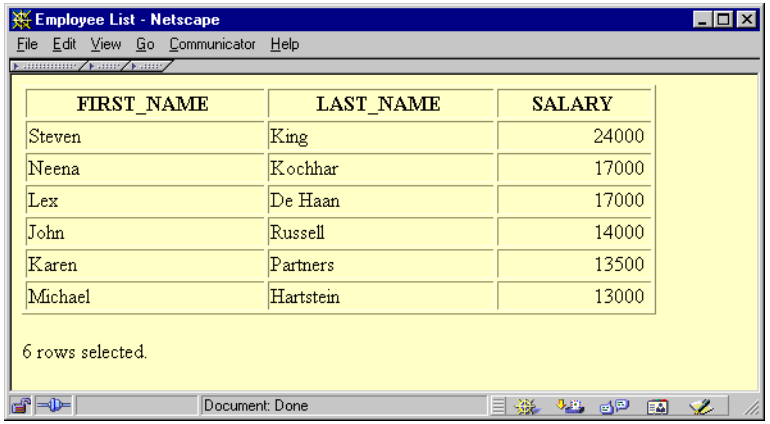
Use the input command to enter the commands necessary to set up the HTML options and the query you want for your report.

```
INPUT
SET MARKUP HTML ON SPOOL ON HEAD '<TITLE>Employee List</title> -
STYLE TYPE="TEXT/CSS"><!--BODY {background: ffffc6} --></STYLE>'
SET ECHO OFF
SPOOL employee.htm
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM EMP_DETAILS_VIEW
WHERE SALARY>12000;
SPOOL OFF
SET MARKUP HTML OFF
SET ECHO ON
.
```

As this script contains SQL*Plus commands, do not attempt to run it with / (slash) from the buffer because it will fail. Save the script and use START to execute it:

```
SAVE employee.sql
START employee.sql
```

As well as writing the html spool file, *employee.htm*, the output is also displayed on screen because SET TERMOUT defaults to ON. You can view the spool file, *employee.htm*, in your web browser. It should appear something like the following:



SERVEROUTPUT

To enable the display of text within a PL/SQL block using DBMS_OUTPUT.PUT_LINE, enter

```
SET SERVEROUTPUT ON
```

The following example shows what happens when you execute an anonymous procedure with SET SERVEROUTPUT ON:

```
BEGIN
  DBMS_OUTPUT.PUT_LINE('Task is complete');
END;
/
Task is complete.
PL/SQL procedure successfully completed.
```

The following example shows what happens when you create a trigger with SET SERVEROUTPUT ON:

```
CREATE TRIGGER SERVER_TRIG BEFORE INSERT OR UPDATE -
OR DELETE
ON SERVER_TAB
BEGIN
DBMS_OUTPUT.PUT_LINE('Task is complete.');
END;
/
Trigger created.

INSERT INTO SERVER_TAB VALUES ('TEXT');
```

```
Task is complete.
1 row created.
```

To set the output to WORD_WRAPPED, enter

```
SET SERVEROUTPUT ON FORMAT WORD_WRAPPED
SET LINESIZE 20
BEGIN
  DBMS_OUTPUT.PUT_LINE('If there is nothing left to do');
  DBMS_OUTPUT.PUT_LINE('shall we continue with plan B?');
END;
/
```

```
If there is nothing
left to do
shall we continue
with plan B?
```

To set the output to TRUNCATED, enter

```
SET SERVEROUTPUT ON FORMAT TRUNCATED
SET LINESIZE 20
BEGIN
  DBMS_OUTPUT.PUT_LINE('If there is nothing left to do');
  DBMS_OUTPUT.PUT_LINE('shall we continue with plan B?');
END;
/
```

```
If there is nothing
shall we continue wi
```

SHIFTINOUT

To enable the display of shift characters on a terminal that supports them, enter

```
SET SHIFTINOUT VISIBLE
SELECT LAST_NAME, JOB_ID FROM EMP_DETAILS_VIEW
WHERE SALARY > 12000;
```

```
LAST_NAME      JOB_ID
---------- ----------
:JJOO:     :AABBCC:
:AA:abc    :DDEE:e
```

where ":" = visible shift character

uppercase represents multibyte characters

lowercase represents singlebyte characters

SQLBLANKLINES

To allow blank lines in a SQL statement, enter

```
SET SQLBLANKLINES ON
REM Using the SQLTERMINATOR (default is ";")
REM Could have used the BLOCKTERMINATOR (default is ".")
SELECT *

FROM

DUAL

;
```

The following output results:

```
D
-
X
```

SQLCONTINUE

To set the SQL*Plus command continuation prompt to an exclamation point followed by a space, enter

```
SET SQLCONTINUE '! '
```

SQL*Plus will prompt for continuation as follows:

```
TTITLE 'MONTHLY INCOME' -
! RIGHT SQL.PNO SKIP 2 -
! CENTER 'PC DIVISION'
```

The default continuation prompt is "> ".

SQLPROMPT

You need the Select Any Table privilege to successfully run the following example scripts.

To set the SQL*Plus command prompt to show your SID, enter

```
SET TERMOUT OFF
COLUMN X NEW_VALUE Y
SELECT RTRIM(INSTANCE, CHR(0)) X FROM V$THREAD;
SQLPROMPT '&Y SQL>'
SET TERMOUT ON
```

To set the SQL*Plus command prompt to show the current user, enter

```
SET TERMOUT OFF
COLUMN D22 NEW_VALUE VAR
SELECT USERNAME D22 FROM USER_USERS;
SQLPROMPT '&&VAR>'
SET TERMOUT ON
```

These settings are not dynamic. You need to reset them whenever you change instances, such as when you use the connect command to log on to another instance.

SUFFIX

To change the default command-file extension from the default, *.SQL* to *.UFI*, enter

```
SET SUFFIX UFI
```

If you then enter

```
GET EXAMPLE
```

SQL*Plus will look for a file named EXAMPLE.UFI instead of EXAMPLE.SQL.

SHOW

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 | DIMENSION | JAVA CLASS } [*schema.*]*name*]
LNO
PARAMETERS [*parameter_name*]
PNO
REL[EASE]
REPF[OOTER]
REPH[EADER]
SGA
SPOO[L]
SQLCODE
TTI[TLE]
USER

Shows the value of a SQL*Plus system variable or the current SQL*Plus environment.

Terms

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 and SGA, in alphabetical order.

BTI[TLE]

Shows the current BTITLE definition.

ERR[ORS] [{FUNCTION|PROCEDURE|PACKAGE|PACKAGE BODY|TRIGGER |VIEW|TYPE|TYPE BODY | DIMENSION | JAVA CLASS} [*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, type body, dimension, or java class) 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:

```
SHOW PARAMETERS COUNT
```

```
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.

Your output may vary depending on the version and configuration of the Oracle database server to which you are connected. You need SELECT ON V_$PARAMETER object privileges to use the PARAMETERS clause, otherwise you will receive a message

```
ORA-00942: table or view does not exist
```

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. Note, you need SELECT ON V_$SGA object privileges to use the SGA clause, otherwise you will receive a message

```
ORA-00942: table or view does not exist
```

SQLCODE

Shows the value of SQL.SQLCODE (the SQL return code of the most recent operation).

TTI[TLE]

Shows the current TTITLE definition.

USER

Shows the username you are currently using to access SQL*Plus. If you connect as "/ AS SYSDBA", then the SHOW USER command displays

```
USER is "SYS"
```

Examples

To list the current LINESIZE, enter

```
SHOW LINESIZE
```

If the current linesize is 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:

```
CONNECT SYSTEM/MANAGER
CREATE PROCEDURE HR.PROC1 AS
BEGIN
:P1 := 1;
END;
/
```

```
Warning: Procedure created with compilation errors.
```

```
SHOW ERRORS
```

```
Errors for PROCEDURE HR.PROC1:
LINE/COL ERROR
--------------------------------------------------------
3/1      PLS-00049: bad bind variable 'P1'
```

```
SHOW ERRORS PROCEDURE PROC1
NO ERRORS.
SHOW ERRORS PROCEDURE HR.PROC1
Errors for PROCEDURE HR.PROC1:
LINE/COL ERROR
--------------------------------------------------------
3/3      PLS-00049: bad bind variable 'P1'
```

To show whether AUTORECOVERY is enabled, enter

```
SHOW AUTORECOVERY
```

```
AUTORECOVERY ON
```

To display the connect identifier for the default instance, enter

```
SHOW INSTANCE
```

```
INSTANCE "LOCAL"
```

To display the location for archive logs, enter

```
SHOW LOGSOURCE
```

```
LOGSOURCE "/usr/oracle90/dbs/arch"
```

To display information about the SGA, enter

```
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

Syntax

SHUTDOWN [ABORT|IMMEDIATE|NORMAL|TRANSACTIONAL [LOCAL]]

Shuts down a currently running Oracle instance, optionally closing and dismounting a database. You cannot use SHUTDOWN to stop Oracle instances on Oracle7 servers.

Terms

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

ABORT

Proceeds with the fastest possible shutdown of the database without waiting for calls to complete or users to disconnect.

Uncommitted transactions are not rolled back. Client SQL statements currently being processed are terminated. All users currently connected to the database are implicitly disconnected and the next database startup will require instance recovery.

You must use this option if a background process terminates abnormally.

IMMEDIATE

Does not wait for current calls to complete or users to disconnect from the database.

Further connects are prohibited. The database is closed and dismounted. The instance is shutdown and no instance recovery is required on the next database startup.

NORMAL

NORMAL is the default option which waits for users to disconnect from the database.

Further connects are prohibited. The database is closed and dismounted. The instance is shutdown and no instance recovery is required on the next database startup.

TRANSACTIONAL [LOCAL]

Performs a planned shutdown of an instance while allowing active transactions to complete first. It prevents clients from losing work without requiring all users to log off.

No client can start a new transaction on this instance. Attempting to start a new transaction results in disconnection. After completion of all transactions, any client still connected to the instance is disconnected. Now the instance shuts down just as it would if a SHUTDOWN IMMEDIATE statement was submitted. The next startup of the database will not require any instance recovery procedures.

The LOCAL mode specifies a transactional shutdown on the local instance only, so that it only waits on local transactions to complete, not all transactions. This is useful, for example, for scheduled outage maintenance.

Usage

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.

Examples

To shutdown the database in normal mode, enter

```
SHUTDOWN
```

```
Database closed.
Database dismounted.
Oracle instance shut down.
```

SPOOL

Syntax

SPO[OL] [*file_name*[.*ext*]|OFF|OUT]

Stores query results in a file, or optionally sends the file to a printer.

Terms

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. This option is not available on some operating systems.

Enter SPOOL with no clauses to list the current spooling status.

Usage

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

```
SPOOL DIARY
```

To stop spooling and print the file on your default printer, enter

```
SPOOL OUT
```

START

Syntax

STA[RT] {*uri*|*file_name*[.*ext*] } [*arg...*]

Runs the SQL*Plus statements in the specified command file. The command file can be called from the local file system or from a web server. *uri* is only supported on Windows platforms in this release.

Terms

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

uri

Specifies the Uniform Resource Identifier of a script to run on the specified web server. SQL*Plus supports HTTP, FTP and gopher protocols.

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

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 @ ("at" sign) and @@ (double "at" sign) commands in this chapter for further information on these commands.

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

Examples

A file named PROMOTE with the extension SQL, used to promote employees, might contain the following command:

```
SELECT FIRST_NAME, LAST_NAME, JOB_ID, SALARY
FROM EMP_DETAILS_VIEW
WHERE JOB_ID='&1' AND SALARY>&2;
```

To run this command file, enter

```
START PROMOTE ST_MAN 7000
```

or if it is located on a web server, enter a command in the form:

```
START HTTP://HOST.DOMAIN/PROMOTE.SQL ST_MAN 7000
START FTP://HOST.DOMAIN/PROMOTE.SQL  ST_MAN 7000
START GOPHER://HOST.DOMAIN/PROMOTE.SQL ST_MAN 7000
```

Where HOST.DOMAIN must be replaced by the host.domain name for the web server where the script is located.

In either case, SQL*Plus then executes the following command:

```
SELECT LAST_NAME, LAST_NAME
FROM EMP_DETAILS_VIEW
WHERE JOB_ID='ST_MAN' AND SALARY>7000;
```

and displays the results in SQL*Plus.

STARTUP

Syntax

STARTUP [FORCE] [RESTRICT] [PFILE=*filename*] [MOUNT [*dbname*]
| OPEN [*open_options*] [*dbname*] | NOMOUNT]

where *open_options* has the following syntax:

READ {ONLY | WRITE [RECOVER]} | RECOVER

Starts an Oracle instance with several options, including mounting and opening a database. You cannot use STARTUP to start Oracle7 instances.

Terms

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. If PFILE is not specified, then the default STARTUP parameter file is used. The default file used is platform specific. For example, the default file is $ORACLE_HOME/dbs/init$ORACLE_SID.ora on UNIX, and %ORACLE_HOME%\database\initORCL.ora on Windows.

MOUNT *dbname*

Mounts a database but does not open it.

dbname is 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.

OPEN

Mounts and opens the specified database.

NOMOUNT

Causes the database not to be mounted upon instance startup.

Cannot be used with 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.

Usage

You must be connected to a database as SYSOPER, or SYSDBA. You cannot be connected via a multi-threaded server.

STARTUP with no arguments is equivalent to STARTUP OPEN.

STARTUP OPEN RECOVER mounts and opens the database even when recovery fails.

Examples

To start an instance using the standard parameter file, mount the default database, and open the database, enter

```
STARTUP
```

or enter

```
STARTUP OPEN database
```

To start an instance using the standard parameter file, mount the default database, and open the database, enter

```
STARTUP FORCE RESTRICT NOMOUNT
```

To start an instance using the parameter file TESTPARM without mounting the database, enter

```
STARTUP PFILE=testparm NOMOUNT
```

To shutdown a particular database, immediately restart and open it, allow access only to database administrators, and use the parameter file MYINIT.ORA. enter

```
STARTUP FORCE RESTRICT PFILE=myinit.ora OPEN database
```

To startup an instance and mount but not open a database, enter

```
CONNECT / as SYSDBA
```

```
Connected to an idle instance.
```

```
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

Syntax

STORE SET *file_name*[.*ext*] [CRE[ATE]|REP[LACE]|APP[END]]

Saves attributes of the current SQL*Plus environment in a host operating system file (a command file).

Terms

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

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.

Examples

To store the current SQL*Plus system variables in a file named DEFAULTENV with the default command-file extension, enter

```
STORE SET DEFAULTENV
```

To append the current SQL*Plus system variables to an existing file called DEFAULTENV with the extension OLD, enter

```
STORE SET DEFAULTENV.OLD APPEND
```

TIMING

Syntax

TIMI[NG] [START *text*|SHOW|STOP]

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.

Terms

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. For other information about TIMING, see SET AUTOTRACE

Usage

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 the SET TIMING command 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

```
TIMING START SQL_TIMER
```

To list the current timer's title and accumulated time, enter

```
TIMING SHOW
```

To list the current timer's title and accumulated time and to remove the timer, enter

```
TIMING STOP
```

TTITLE

Syntax

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

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*

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.

Terms

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 (the current line number), SQL.PNO (the current page number), SQL.RELEASE (the current Oracle release number), SQL.SQLCODE (the current error code), or SQL.USER (the 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 bold 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

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:

```
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

```
TTITLE LEFT 'Monthly Analysis' CENTER '01 Jan 2001' -
RIGHT 'Page:' FORMAT 999 SQL.PNO SKIP CENTER -
'Data in Thousands'
```

```
Monthly Analysis              01 Jan 2001          Page:   1
                              Data in Thousands
```

To suppress the top title display without changing its definition, enter

```
TTITLE OFF
```

UNDEFINE

Syntax

UNDEF[INE] *variable* ...

Deletes one or more user variables that you defined either explicitly (with the DEFINE command) or implicitly (with an argument to the START command).

Terms

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

```
UNDEFINE POS
```

To undefine two user variables named MYVAR1 and MYVAR2, enter

```
UNDEFINE MYVAR1 MYVAR2
```

VARIABLE

Syntax

VAR[IABLE] [*variable* [NUMBER|CHAR|CHAR (*n* [CHAR|BYTE])|NCHAR|NCHAR (*n*) |VARCHAR2 (*n* [CHAR|BYTE])|NVARCHAR2 (*n*)|CLOB|NCLOB|REFCURSOR]]

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.

Terms

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 fixed length.

CHAR

Creates a variable of type CHAR (character) with length one.

CHAR (*n*[CHAR|BYTE])

Creates a variable of type CHAR having a length of *n* bytes or *n* characters. The maximum that *n* can be is 2000 bytes, and the minimum is 1 byte or 1 character. The maximum *n* for a CHAR variable with character semantics is determined by the number of bytes required to store each character for the chosen character set, with an upper limit of 2000 bytes. The length semantics are determined by the length qualifiers CHAR or BYTE, and if not explicitly stated, the value of the NLS_LENGTH_SEMANTICS environment variable is applied to the bind variable. Explicitly stating the length semantics at variable definition stage will always take precedence over the NLS_LENGTH_SEMANTICS setting.

NCHAR

Creates a variable of type NCHAR (national character) with length one.

NCHAR (*n*)

Creates a variable of type NCHAR having a length of *n* characters. The maximum that *n* can be is determined by the number of bytes required to store each character for the chosen national character set, with an upper limit of 2000 bytes. The only exception to this is when a SQL*Plus session is connected to a pre Oracle9*i* server, or the SQLPLUSCOMPATIBILITY system variable is set to a version less than 9.0.0. In this case the length *n* can be in bytes or characters depending on the chosen national character set, with the upper limit of 2000 bytes still retained.

VARCHAR2 (*n*[CHAR|BYTE])

Creates a variable of type VARCHAR2 having a length of up to *n* bytes or *n* characters. The maximum that *n* can be is 4000 bytes, and the minimum is 1 byte or 1 character. The maximum *n* for a VARCHAR2 variable with character semantics is determined by the number of bytes required to store each character for the chosen character set, with an upper limit of 4000 bytes. The length semantics are determined by the length qualifiers CHAR or BYTE, and if not explicitly stated, the value of the NLS_LENGTH_SEMANTICS environment variable is applied to the bind variable. Explicitly stating the length semantics at variable definition stage will always take precedence over the NLS_LENGTH_SEMANTICS setting.

NVARCHAR2 (*n*)

Creates a variable of type NVARCHAR2 having a length of up to *n* characters. The maximum that *n* can be is determined by the number of bytes required to store each character for the chosen national character set, with an upper limit of 4000 bytes. The only exception to this is when a SQL*Plus session is connected to a pre Oracle9*i* server, or the SQLPLUSCOMPATIBILITY system variable is set to a version less than 9.0.0. In this case the length *n* can be in bytes or characters depending on the chosen national character set, with the upper limit of 4000 bytes still retained.

CLOB

Creates a variable of type CLOB.

NCLOB

Creates a variable of type NCLOB.

REFCURSOR

Creates a variable of type REF CURSOR.

Usage

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:

```
VARIABLE id NUMBER
BEGIN
  :id := EMP_MANAGEMENT.HIRE
  ('BLAKE','MANAGER','KING',2990,'SALES');
END;
/
```

The value returned by the stored procedure is being placed in the bind variable, :id. It can be displayed with the PRINT command or used in subsequent PL/SQL subprograms.

The following example illustrates automatically displaying a bind variable:

```
SET AUTOPRINT ON
VARIABLE a REFCURSOR
BEGIN
  OPEN :a FOR SELECT LAST_NAME, CITY, DEPARTMENT_ID
  FROM EMP_DETAILS_VIEW
  WHERE SALARY > 12000
  ORDER BY DEPARTMENT_ID;
END;
/
```

```
PL/SQL procedure successfully completed.

LAST_NAME                 CITY                           DEPARTMENT_ID
------------------------- ------------------------------ -------------
Hartstein                 Toronto                                   20
Russell                   Oxford                                    80
Partners                  Oxford                                    80
King                      Seattle                                   90
Kochhar                   Seattle                                   90
De Haan                   Seattle                                   90

6 rows selected.
```

In the above example, there is no need to issue a PRINT command to display the variable.

The following example creates some variables:

```
VARIABLE id NUMBER
VARIABLE txt CHAR (20)
VARIABLE myvar REFCURSOR
```

Enter VARIABLE with no arguments to list the defined variables:

```
VARIABLE
```

```
variable id
datatype NUMBER

variable txt
datatype CHAR(20)

variable myvar
datatype REFCURSOR
```

The following example lists a single variable:

```
VARIABLE txt
```

```
variable txt
datatype CHAR(20)
```

The following example illustrates producing a report listing individual salaries and computing the departmental salary cost for employees who earn more than $12,000 per month:

```
VARIABLE rc REFCURSOR
BEGIN
  OPEN :rc FOR SELECT DEPARTMENT_NAME, LAST_NAME, SALARY
  FROM EMP_DETAILS_VIEW
  WHERE SALARY > 12000
  ORDER BY DEPARTMENT_NAME, LAST_NAME;
END;
/
```

```
PL/SQL procedure successfully completed.
```

```
SET PAGESIZE 100 FEEDBACK OFF
TTITLE LEFT '*** Departmental Salary Bill ***' SKIP 2
COLUMN SALARY FORMAT $999,990.99 HEADING 'Salary'
COLUMN DEPARTMENT_NAME HEADING 'Department'
COLUMN LAST_NAME HEADING 'Employee'
```

```
COMPUTE SUM LABEL 'Subtotal:' OF SALARY ON DEPARTMENT_NAME
COMPUTE SUM LABEL 'Total:' OF SALARY ON REPORT
BREAK ON DEPARTMENT_NAME SKIP 1 ON REPORT SKIP 1
PRINT rc
```

```
*** Departmental Salary Bill ***

DEPARTMENT_NAME                Employee                       Salary
------------------------------ ------------------------ ------------
Executive                      De Haan                    $17,000.00
                               King                       $24,000.00
                               Kochhar                    $17,000.00
******************************                           ------------
Subtotal:                                                 $58,000.00

Marketing                      Hartstein                  $13,000.00
******************************                           ------------
Subtotal:                                                 $13,000.00

Sales                          Partners                   $13,500.00
                               Russell                    $14,000.00
******************************                           ------------
Subtotal:                                                 $27,500.00


                                                         ------------
Total:                                                    $98,500.00
```

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

```
VARIABLE T CLOB
BEGIN
  SELECT CLOB_COL INTO :T FROM CLOB_TAB;
END;
/
```

```
PL/SQL PROCEDURE SUCCESSFULLY COMPLETED
```

To print 200 characters from the column clob_col, enter

```
SET LINESIZE 70
SET LONG 200
PRINT T
```

```
T
--------------------------------------------------------------------------
Remember to run the Departmental Salary Bill report each month. This r
eport contains confidential information.
```

To set the printing position to the 21st character, enter

```
SET LOBOFFSET 21
PRINT T
```

```
T
--------------------------------------------------------------------------
Departmental Salary Bill report each month. This report contains confi
dential information.
```

For more information on creating CLOB columns, see your *Oracle9i SQL Reference*.

WHENEVER OSERROR

Syntax

WHENEVER OSERROR
{EXIT [SUCCESS|FAILURE|*n*|*variable*|*:BindVariable*] [COMMIT|ROLLBACK]
|CONTINUE [COMMIT|ROLLBACK|NONE]}

Exits SQL*Plus if an operating system error occurs (such as a file I/O error).

Terms

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

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:

```
WHENEVER OSERROR EXIT
START no_such_file
```

```
OS Message: No such file or directory
Disconnected from Oracle......
```

WHENEVER SQLERROR

Syntax

WHENEVER SQLERROR
{EXIT [SUCCESS|FAILURE|WARNING|*n*|*variable*|*:BindVariable*]
[COMMIT|ROLLBACK]|CONTINUE [COMMIT|ROLLBACK|NONE]}

Exits SQL*Plus if a SQL command or PL/SQL block generates an error.

Terms

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

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:

```
WHENEVER SQLERROR EXIT SQL.SQLCODE
UPDATE EMP_DETAILS_VIEW SET SALARY = SALARY*1.1
```

The following SQL command error causes SQL*Plus to exit and return the SQL error code:

```
WHENEVER SQLERROR EXIT SQL.SQLCODE
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:

```
DEFINE MY_ERROR_VAR = 99
WHENEVER SQLERROR EXIT my_error_var
UPDATE non_existed_table set col1 = col1 + 1;
```

```
UPDATE NON_EXISTED_TABLE set col1 = col1 + 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:

```
WHENEVER SQLERROR EXIT SQL.SQLCODE
COLUMN LAST_NAME HEADIING "EMPLOYEE NAME"
```

```
Unknown COLUMN option "HEADIING"
```

```
SHOW non_existed_option
```

```
Unknown SHOW option "NON_EXIST_OPTION"
GET non_existed_file.sql
```

```
Unable to open "NON_EXIST_FILE.SQL"
```

The following PL/SQL block error causes SQL*Plus to exit and return the SQL error code:

```
WHENEVER SQLERROR EXIT SQL.SQLCODE
BEGIN
  SELECT COLUMN_DOES_NOT_EXIST FROM DUAL;
END;
/
```

```
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.....
```

A

SQL*Plus Error Messages

This appendix lists error messages generated by SQL*Plus. For error messages generated by Oracle, refer to the *Oracle9i Error Messages*.

This chapter contains information about:

- SQL*Plus Error Messages and Codes
- COPY Command Messages

SQL*Plus Error Messages and Codes

SP2-0002 ACCEPT statement must specify a variable name.

Cause: Required variable name was missing after the ACCEPT command.

Action: Re-enter the ACCEPT command with a variable argument to store the input value.

SP2-0003 Ill-formed ACCEPT command starting as *command_string*

Cause: An invalid option was used in the ACCEPT command.

Action: Check the syntax of the ACCEPT command in Chapter 8, "Command Reference" for the correct option.

SP2-0004 Nothing to append

Cause: There was no specified text entered after the APPEND command.

Action: Re-enter the APPEND command with the specified text.

SP2-0006 Not enough room to format computations

Cause: Unable to allocate memory to format computations.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0015 No break(s) defined

Cause: There was no break defined.

Action: Define a break. Check the syntax of the BREAK command in Chapter 8, "Command Reference" for the correct options.

SP2-0016 Break specification must start with ON/BY or ACROSS keyword

Cause: An invalid option was used in the BREAK command.

Action: Check the syntax of the BREAK command in Chapter 8, "Command Reference" for the correct options.

SP2-0017 Missing column name after '*keyword_name*' keyword

Cause: There was no column name after the specified keyword.

Action: Enter a column name after the specified keyword.

SP2-0019 Invalid numeric argument to *option_name* option

Cause: An invalid numeric argument was used in the specified option.

Action: Correct the argument and try again.

SP2-0020 No storage available for *column_name*

Cause: An error has occurred. SQL*Plus was unable to allocate memory for a BREAK command.

Action: Allocate more memory by closing some applications.

SP2-0022 Cannot allocate space to modify the *buffer_name* buffer variable

Cause: An internal error occurred.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0023 String not found

Cause: The search string specified was not found.

Action: Check the search string to make sure that it is valid.

SP2-0024 Nothing to change

Cause: There was nothing in the SQL buffer when using the CHANGE command.

Action: Make sure the SQL buffer is not empty before using the CHANGE command.

SP2-0025 Invalid change string

Cause: An invalid option was used in the CHANGE command.

Action: Check the syntax of the CHANGE command in Chapter 8, "Command Reference" for the correct options.

SP2-0026 No lines to delete

Cause: There was nothing in the SQL buffer when using the DEL command.

Action: Make sure the SQL buffer is not empty before using the DEL command.

SP2-0027 Input is too long (> *max_characters* characters) - line ignored

Cause: The input value specified was too long.

Action: Re-enter with fewer characters.

SP2-0028 INTERNAL SQL*Plus ERROR - Invalid mode (*mode_number*)

Cause: An internal error occurred.

Action: Note the message and number, and contact Oracle Support Services.

SP2-0029 Command buffer space exhausted

Cause: A large SQL or PL/SQL script is being executed from SQL*Plus.

Action: Reduce the size of the SQL statement or PL/SQL block by one of the following:

- Remove extra white space and comments.
- Re-code to use fewer commands and/or shorter variable names.
- Place sections of the block into stored (or packaged) procedures, and then call these procedures from the block.

SP2-0030 No room for another line

Cause: The maximum number of lines in a SQL statement or PL/SQL block has been exceeded.

Action: Reduce the number of lines and try again.

SP2-0038 Command too long. (*max_characters* characters)

Cause: The specified command entered was too long.

Action: Check Chapter 8, "Command Reference" for the limitation.

SP2-0039 Command line overflow while substituting into *command_name*

Cause: The maximum length of the command line has been exceeded.

Action: Reduce the length of the substitution string.

SP2-0042 Unknown command *command_name* - rest of line ignored

Cause: The command entered was not valid.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct options of the command you used.

SP2-0044 For a list of known commands enter HELP and to leave enter EXIT

Cause: An unknown command was entered.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct options of the command you used.

SP2-0045 No *column_name* defined

Cause: No columns have been defined.

Action: No action required.

SP2-0046 *column_name* not defined

Cause: The column name specified was not defined.

Action: Retry again with a valid column name.

SP2-0047 Invalid number for *option_name* option

Cause: An invalid number was used for this option.

Action: Re-try the operation with a valid number.

SP2-0051 Switch value is *switch_value* and is not handled properly

Cause: An internal error occurred.

Action: Note the message and number, and contact Oracle Support Services.

SP2-0052 Like *column_name, column_name* not defined

Cause: The column which the format is based on was not defined.

Action: Use the COLUMN command to make sure the column the format is based on is defined first.

SP2-0054 No room to allocate *definition_name* definition. Ignored

Cause: Unable to allocate memory to process the COLUMN command.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0055 Out of room while allocating portion of new *definition_name*. Old definition (if any) retained

Cause: Unable to allocate memory to store the new definition.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0080 No COMPUTES currently defined

Cause: No COMPUTE definition.

Action: Define a COMPUTE. Check the syntax of the COMPUTE command in Chapter 8, "Command Reference" for the correct options.

SP2-0081 Maximum of *number* COMPUTE functions allowed at a time

Cause: The maximum of COMPUTE functions has been exceeded.

Action: Reduce the number of COMPUTE functions.

SP2-0082 No COMPUTE functions requested

Cause: No COMPUTE functions requested.

Action: No action required.

SP2-0083 Warning: COMPUTE option *function_name* specified *number* times

Cause: A label or a function was specified more than once.

Action: Remove the unnecessary labels or functions.

SP2-0084 COMPUTE ON keyword specified already

Cause: The ON keyword was specified more than once.

Action: Specify the ON keyword once in the command.

SP2-0085 COMPUTE OF keyword specified already

Cause: The OF keyword was specified more than once.

Action: Specify the OF keyword once in the command.

SP2-0087 No room to allocate COMPUTE control block for *column_name*

Cause: Unable to allocate memory to process the COMPUTE command.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0088 Missing *keyword_name* keyword.
Usage: STORE [SET *filename*[.*ext*] [CRE[ATE]|REP[LACE]|APP[END]]

Cause: Missing a keyword in the statement.

Action: Check the syntax in Chapter 8, "Command Reference" for the options of the command you used, and use the keyword in the appropriate place.

SP2-0092 Missing columns for *keyword_name* keyword

Cause: The column name was not been specified for the keyword.

Action: Specify the column name and try again.

SP2-0096 No more room to allocate INTO variable *variable_name*

Cause: Unable to allocate memory to process the COMPUTE command.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0097 No storage to allocate ON column *column_name*

Cause: Unable to allocate memory to process the COMPUTE command.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0098 No storage to allocate COMPUTE block for *column_name*

Cause: Unable to allocate memory to process the COMPUTE command.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0103 Nothing in SQL buffer to run

Cause: Nothing was in the SQL buffer to run.

Action: Enter a valid SQL command.

SP2-0105 Illegal, or missing, entity name

Cause: File name was not specified in the GET or SAVE commands.

Action: Specify a file name and try again.

SP2-0107 Nothing to save

Cause: Nothing in the SQL buffer when attempting to save the content to a file.

Action: Enter a SQL command to save.

SP2-0108 The names CREATE, REPLACE, APPEND, and abbreviations may not be used

Cause: The file name specified was the word "file".

Action: Put the name in single quotes.

SP2-0109 Cannot append to file *file_name*

Cause: An attempt was made to append the content of the SQL buffer to a file and the file could not be written. Possible causes:

- An error was encountered when creating the destination file.
- A directory name specified in the SAVE statement was not found.
- A system error made it impossible to open the file.

Action: Take the following actions

- Check that the destination is valid and that there is sufficient space on the destination device.
- Check the statement for a typing mistake in the directory name. Then issue the statement again after correcting the directory name.

SP2-0110 Cannot create save file *file_name*

Cause: An attempt was made to save the content of the SQL buffer to a file and the file could not be written. Possible causes:

- An error was encountered when creating the destination file.
- A directory name specified in the SAVE statement was not found.
- A system error made it impossible to open the file.

Action: Take the following actions:

- Check that the destination is valid and that there is sufficient space on the destination device.
- Check the statement for a typing mistake in the directory name. Then issue the statement again after correcting the directory name.

SP2-0111 Cannot close save file *file_name*

Cause: The file was in use.

Action: Release the file from the other process.

SP2-0116 Illegal SAVE command

Cause: An invalid option was used in the SAVE command.

Action: Check the syntax of the SAVE command in Chapter 8, "Command Reference" for the correct options.

SP2-0134 No symbols currently defined

Cause: No DEFINE symbols were defined.

Action: No action required.

SP2-0135 Symbol *symbol_name* is UNDEFINED

Cause: The specified symbol was undefined.

Action: Re-enter the DEFINE command with an assignment clause or a valid symbol or variable name.

SP2-0136 DEFINE requires an equal sign (=)

Cause: Expecting an equal sign after a symbol or variable name in the DEFINE command.

Action: Specify an equal sign after the symbol or variable name.

SP2-0137 DEFINE requires a value following equal sign

Cause: There was no value for the variable or symbol. SQL*Plus expected a value to be assigned to a symbol or variable name after the equal sign.

Action: Specify a value for the symbol or variable.

SP2-0138 DEFINE *variable* not added (no room)

Cause: Maximum number of variables that can be defined in a SQL*Plus session was exceeded.

Action: UNDEFINE any unused variables to make room for this variable and re-run the command.

SP2-0145 Udalnk is not 12345. Probably a link error

Action: The SQL*Plus executable is not linked correctly.

Action: Make a note of the message and the number, then contact the System Administrator to re-link SQL*Plus.

SP2-0146 Unable to allocate dynamic space needed (*number_of_bytes* bytes) - exiting

Cause: An internal error occurred.

Action: Note the message and number, and contact the System Administrator.

SP2-0152 ORACLE may not be functioning properly

Cause: Unable to initialize a session to the Oracle instance.

Action: Make a note of the message and the number, then contact Database Administrator.

SP2-0157 Unable to CONNECT to ORACLE after 3 attempts, exiting SQL*Plus

Cause: Unable to connect to Oracle after three attempts.

Action: Validate login details and re-try.

SP2-0158 Unknown *command_name* option "*option_name*"
Usage: SET SHIFT[INOUT] [VIS[IBLE|INV[ISIBLE]]
SET NEWP[AGE] [1 | n | NONE]

Cause: An invalid option was specified for the given command.

Action: Check the syntax of the command in Chapter 8, "Command Reference" for the correct options.

SP2-0160 Unable to open *file_name*

Cause: Possible causes:

- The file was not found under the specified name in the specified location.
- File lacked the necessary privileges to open the file.
- A system error made it impossible to open the file.

Action: Take the following actions:

- Make sure the file name specified is stored in the appropriate directory.
- Make sure that the file has the privileges necessary for access. If it does not then change privileges accordingly.
- Consult operating system documentation or contact the System Administrator.

SP2-0161 Line *line_number* truncated

Cause: The line in the file was too long.

Action: No action required or reduce the length of the line.

SP2-0162 Unable to close *file_name*

Cause: Unable to close the specified file as it was being used.

Action: Release the file from the other process.

SP2-0171 HELP not accessible

Cause: On-line SQL*Plus help is not installed in this Oracle instance.

Action: Contact the Database Administrator to install the on-line help.

SP2-0172 No HELP available

Cause: There is no help information available for the specified command.

Action: Contact the Database Administrator to install the help system.

SP2-0176 Option ? Is invalid

Cause: The option ? is not a valid in this command.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct options for the command you used.

SP2-0187 Error in variable assignment

Cause: The assignment for the specified variable was incorrect.

Action: Refer to the ACCEPT command in Chapter 8, "Command Reference" and correct the syntax.

SP2-0223 No lines in *buffer_name* buffer

Cause: There are no lines stored in the buffer.

Action: Enter SQL statements into the buffer.

SP2-0224 Invalid starting line number

Cause: The line number specified was incorrect.

Action: Check and make sure that the line number is correct and try again.

SP2-0225 Invalid ending line number

Cause: The line number specified was incorrect.

Action: Check and make sure that the line number is correct and try again.

SP2-0226 Invalid line number *current_line_number*

Cause: Invalid line number was specified.

Action: Re-enter with a valid line number.

SP2-0232 Input too long. Must be less than *number_of_characters* characters

Cause: The input value was too long.

Action: Reduce the size of the value and re-enter.

SP2-0233 Unable to obtain userid after *number_of_attempts* attempts. Retry command

Cause: SQL*Plus was unable to login after three attempts.

Action: Make sure the userid and password is correct and try again.

SP2-0240 Enter value for *variable_name*:

Cause: A substitution variable was used and SQL*Plus was unable to find a value for that variable.

Action: Enter a value at the prompt for the substitution variable.

SP2-0241 No room for symbol *symbol_name*:(not defined)

Cause: Unable to allocate memory for the symbol.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0244 Cannot issue a PRINT command within a PAGE break

Cause: The PRINT command is not allowed within a PAGE break.

Action: Check Chapter 8, "Command Reference" for the correct syntax.

SP2-0245 Unable to allocate temporary storage for printing

Cause: Unable to allocate temporary storage for printing.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0246 Illegal FORMAT string *column_ format_name*

Cause: An invalid format was specified for the column.

Action: Specify a valid format for the column.

SP2-0249 *variable_name* not a valid variable type for printing

Cause: The specified variable is not valid for printing.

Action: Check the variable type before re-typing the command.

SP2-0253 Data item *line_number* (*data_item_name*) will not fit on line

Cause: The current line size setting is too small to fit the specified data item on a line.

Action: Increase the line size so that the item can be displayed.

SP2-0258 Could not create variable *variable_name* for column *column_name*

Cause: The specified variable could not be created for column – internal error or out of memory.

Action: Check memory usage.

SP2-0259 Could not create variable *variable_name* for COMPUTE INTO

Cause: The specified variable could not be created.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct option of the command you used.

SP2-0260 Computation for column *column_name* not uniquely qualified. Could be for table *table_name* or *table_name*. Computation ignored

Cause: The specified column was not uniquely qualified in the statement.

Action: Check syntax in Chapter 8, "Command Reference" for the correct option of the command you used.

SP2-0262 No room to allocate CCBDEF pointer array

Cause: An internal memory error occurred.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0263 No room to allocate COMPUTE block for *column_name* ON *page/report/column_name*

Cause: Insufficient memory allocated to the COMPUTE block.

Action: Allocate more memory by closing other applications.

SP2-0265 *option_name* must be set ON or OFF

Cause: An invalid SET option name was specified.

Action: Re-enter with either ON or OFF as one of the SET options.

SP2-0266 Internal error: buffer (*buffer_size*) smaller than l (*buffer_limit*)

Cause: An internal error occurred.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0267 *option_name* option *parameter_number* (*lower_range* through *upper_range*)

Cause: A value for a parameter was out of the specified range.

Action: Check the limits of the parameter and enter a value that is within the limit.

SP2-0268 *option_name* option not a valid number

Cause: Non-numeric value (integer) was entered for a parameter.

Action: Enter a valid numeric value (integer).

SP2-0271 *variable_name* is not a buffer variable

Cause: The specified variable was not defined as a buffer.

Action: Make sure that the buffer variable name is correct and try again.

SP2-0272 *character_name* character cannot be alphanumeric or white-space

Cause: The specified character in the SET command cannot be alphanumeric or white-space.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct option of the command you used.

SP2-0277 *entered_value* value not valid

Cause: The value entered was incorrect.

Action: Re-enter with a valid value.

SP2-0281 *option_name* missing set option
Usage: SET SHIFT[INOUT] [VIS[IBLE|INV[ISIBLE]]
SET MARKUP HTML [ON|OFF] [HEAD *text*] [BODY *text*] [TABLE *text*]
[ENTMAP [ON|OFF]] [SPOOL [ON|OFF]] [PRE[FORMAT] [ON|OFF]]
[-M[ARKUP] \"HTML [ON|OFF] [HEAD *text*] [BODY *text*]

Cause: SET option was missing in the command.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct options of the command you used.

SP2-0306 Invalid option

Cause: Invalid option was specified for the command.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct options of the command you used.

SP2-0308 Cannot close spool file

Cause: The file is currently being used.

Action: Release the file from the other process.

SP2-0309 SQL*Plus command procedures may only be nested to a depth of *number_of_nested_procedures*

Cause: Maximum number of nested procedures or scripts was reached.

Action: Reduce the number of nested procedures or scripts.

SP2-0310 Unable to open file *file_name*

Cause: Unable to open the specified file.

Action: Check and make sure the file name is valid.

SP2-0311 String expected but not found

Cause: SQL*Plus was expecting a string at the end of the command, but could not find it.

Action: Retry the command with a valid string. Check the syntax in the Chapter 8, "Command Reference" for the correct options for the command you used.

SP2-0312 Missing terminating quote (quote_type)

Cause: The DESCRIBE command schema or object did not have a terminating quote.

Action: Close the opening quotation mark with the corresponding closing quotation mark.

SP2-0317 Expected symbol name is missing

Cause: SQL*Plus was expecting a symbol, but it was not specified.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct options for the command you used.

SP2-0318 Symbol name beginning *variable_name*.. Is too long (max *max_name_length*)

Cause: Specified variable name exceeded the maximum name length.

Action: Reduce the size of the symbol name and re-enter.

SP2-0323 No room to add timing element - request denied

Cause: Unable to allocate memory while trying to run the TIMING command.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0324 Operating system timing error *error_option_number* - request denied

Cause: The TIMING command failed to initialize due to a possible operating system error.

Action: Resolve the operating system error and try again.

SP2-0325 No timing elements to *option_name*

Cause: There are no timers recorded to SHOW or STOP.

Action: Check that timers were created with the TIMING command.

SP2-0328 No room to allocate title buffer

Cause: Unable to allocate memory while trying to run the TTITLE or BTITLE command.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0332 Cannot create spool file

Cause: Possible causes:

- Insufficient privileges to create a file.
- A system error made it impossible to create a file.

Action: Take the following actions:

- Change privileges to allow creation of the file.
- Consult the operating system documentation or contact the System Administrator.

SP2-0333 Illegal spool file name: "*spool_name* " (bad character: '*character_name*')

Cause: An invalid filename was entered in the SPOOL command.

Action: Correct the filename and re-enter.

SP2-0341 Line overflow during variable substitution (>*number_of_characters* characters at line *line_number*)

Cause: The maximum number of characters was exceeded in the SQL buffer after the substitution variable has been expanded.

Action: Reduce the length in the substitution variable and try again.

SP2-0357 Out of temporary storage

Cause: Unable to allocate memory while trying to run the command.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0359 Memory exhausted

Cause: Unable to allocate memory while trying to run the command.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0381 *command_name* is not available

Cause: The command specified is not implemented.

Action: Use the appropriate SQL*Plus command. Refer to Chapter 8, "Command Reference" for a list of commands and their correct syntax.

SP2-0382 The *command_name* command is not available

Cause: The command was not recognized, or it is disabled. This occurs if it is a command that does not have any meaning in SQL*Plus (such as a SQL buffer editing command), or it is not allowed for security reasons.

Action: Remove the command from the script. Refer to Chapter 8, "Command Reference" for a list of commands and their correct syntax.

SP2-0392 Cannot UNDEFINE the current edit buffer

Cause: The current edit buffer cannot be undefined.

Action: No action required.

SP2-0394 Illegal buffer name: *buffer_name*

Cause: An buffer name contained an illegal character, for example hyphen (-).

Action: Correct and remove the illegal character from the buffer name.

SP2-0395 Usage: SQLPLUS [[<*option*>] [<*login*>] [<*start*>]]
Where <*option*> ::= -H | -V | [[-M <o>] [-R <n>] [-S]]
-H displays the SQL*Plus version banner and usage syntax
-M <o> uses HTML markup options <o>
-V displays the SQL*Plus version banner
-S uses silent mode
-R <n> uses restricted mode <n>
<*login*> ::= <*username*>[/<*password*>][@<*connect_string*>] | / | /NOLOG
<*start*> ::= @<*filename*>[.<*ext*>] [<*parameter*> ...]

Cause: A SQL*Plus command option was invalid.

Action: Check the syntax for the SQLPLUS command in Chapter 7, "Starting SQL*Plus and Getting Help" for the correct usage.

SP2-0423 Illegal GET command

Cause: An invalid option was used in the GET command.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct options of the command you used.

SP2-0425 *value* is not a valid number

Cause: The value entered in the ACCEPT command was not a number.

Action: Correct the entry and enter a valid number.

SP2-0426 Input truncated to *number_of_characters* characters

Cause: There was no carriage return at the last line of the SQL statement.

Action: Insert a carriage return.

SP2-0450 Usage: WHENEVER SQLERROR.
[CONTINUE [COMMIT | ROLLBACK | NONE] | EXIT [SUCCESS | FAILURE | WARNING | n | <*variable*> | :<*bindvariable*>] [COMMIT | ROLLBACK]]

Cause: An option to WHENEVER SQLERROR was invalid in SQL*Plus.

Action: Specify a valid option.

SP2-0453 Usage: WHENEVER OSERROR
{ CONTINUE [COMMIT | ROLLBACK | NONE] | EXIT [SUCCESS | FAILURE | WARNING | n | <*variable*> | :<*bindvariable*> | OSCODE] [COMMIT | ROLLBACK] }

Cause: An option to WHENEVER OSERROR was invalid in SQL*Plus.

Action: Specify a valid option.

SP2-0480 A missing FROM or TO clause uses the current SQL*Plus connection.
Usage: COPY FROM <*db*> TO <*db*> <*opt*> <*table*> [(<*cols*>)] USING <*sel*>
<*db*> : *database string*, e.g., hr/hr@d:chicago-mktg
<*opt*> : ONE of the keywords: APPEND, CREATE, INSERT or REPLACE.
<*table*>: name of the destination table.
<*cols*> : a comma-separated list of destination column aliases.
<*sel*> : any valid SQL SELECT statement.

Cause: Usage for COPY command was specified incorrectly.

Action: Specify a valid option.

SP2-0495 FROM and TO clauses both missing; specify at least one

Cause: The FROM and TO clauses were missing from the COPY statement.

Action: Specify at least one clause. Check the syntax in Chapter 8, "Command Reference" for the correct options of the command you used.

SP2-0496 Misplaced FROM clause

Cause: The FROM keyword was in the wrong position in the COPY command.

Action: Check the syntax of the COPY command in Chapter 8, "Command Reference" for the correct options.

SP2-0497 Misplaced TO clause

Cause: The TO keyword was in the wrong position in the COPY command.

Action: Check the syntax of the COPY command in Chapter 8, "Command Reference" for the correct options.

SP2-0498 Missing parenthetical column list or USING keyword

Cause: A parenthetical list was missing in the column list or the USING keyword is missing in the COPY command.

Action: Check the syntax of the COPY command in Chapter 8, "Command Reference" for the correct options.

SP2-0499 Misplaced APPEND keyword

Cause: The APPEND keyword was in the wrong position in the COPY command.

Action: Check the syntax of the COPY command in Chapter 8, "Command Reference" for the correct options.

SP2-0501 Error in SELECT statement: *Oracle_database_error_message*

Cause: Invalid SELECT statement found in the COPY command.

Action: Check the syntax of the COPY command in Chapter 8, "Command Reference" for the correct options.

SP2-0513 Misplaced CREATE keyword

Cause: The CREATE keyword was in the wrong position in the COPY command.

Action: Check the syntax of the COPY command in Chapter 8, "Command Reference" for the correct options.

SP2-0514 Misplaced REPLACE keyword

Cause: The REPLACE keyword was in the wrong position in the COPY command.

Action: Check the syntax of the COPY command in Chapter 8, "Command Reference" for the correct options.

SP2-0515 Maximum number of columns (*max_num_columns*) exceeded

Cause: An error occurred with the COPY command, maximum number of columns was exceeded in the command.

Action: Reduce the number of columns and try again.

SP2-0516 Invalid *command_name* name NULL encountered

Cause: Either COLUMN or ATTRIBUTE command used a null column name. Invalid column name specified in the command.

Action: Retry the operation with a valid column name.

SP2-0517 Missing comma or right parenthesis

Cause: A missing command right parenthesis was identified in the COPY command.

Action: Retry the operation with a comma or right parenthesis.

SP2-0518 Missing USING clause

Cause: USING keyword is missing in the USING clause of the COPY command.

Action: Specify the USING keyword before the query statement.

SP2-0519 FROM string missing Oracle Net @database specification

Cause: Missing connect string for the database that contains the data to be copied from in the COPY command.

Action: Re-specify the database. By omitting the FROM clause, the source defaults to the database to which SQL*Plus is connected. Include a FROM clause to specify a source database other than the default.

SP2-0520 TO string missing Net8 Oracle Net @database specification

Cause: Missing connect string for the database containing the destination table in the COPY command.

Action: Re-specify the database. By omitting the TO clause, the source defaults to the database to which SQL*Plus is connected. Include a TO clause to specify a source database other than the default.

SP2-0526 Misplaced INSERT keyword

Cause: The INSERT keyword is misplaced in the COPY command.

Action: Check the syntax of the COPY command in Chapter 8, "Command Reference" for the correct options.

SP2-0540 File *file_name* already exists. Use SAVE *filename*[.*ext*] REPLACE

Cause: The file specified already exists.

Action: Use the REPLACE option to overwrite the existing file, otherwise, specify another file name.

SP2-0545 SET command requires an argument

Cause: An argument was missing in the SET command.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct options of the command you used.

SP2-0546 User requested Interrupt or EOF detected

Cause: Either end-of-file was reached, or CTRL-C was entered to cancel the process.

Action: No action required.

SP2-0547 *option_name* option *value* out of range (*lower_value* through *upper_value*)

Cause: An out of range was reached for the specified SET option.

Action: Check the limits for the option and re-try the operation.

SP2-0548 Usage: VAR[IABLE] [<*variable*> [NUMBER | CHAR | CHAR (n [CHAR|BYTE]) |VARCHAR2 (n [CHAR|BYTE]) | NCHAR | NCHAR (n) |NVARCHAR2 (n) | CLOB | NCLOB | REFCURSOR]]

Cause: Usage message for VARIABLE command.

Action: Check the syntax of the VARIABLE command in Chapter 8, "Command Reference" for the correct usage.

SP2-0549 Usage: PRINT [:<*variable*> ...]

Cause: Usage message for PRINT command.

Action: Check the syntax of the PRINT command in Chapter 8, "Command Reference" for the correct usage.

SP2-0550 Usage: SHOW ERRORS [[FUNCTION | PROCEDURE | PACKAGE | PACKAGE BODY | TRIGGER | VIEW | TYPE | TYPE BODY | JAVA SOURCE | JAVA CLASS] [*schema*.]*name*]

Cause: Usage message for SHOW ERRORS command.

Action: Check the syntax of the SHOW ERRORS command in Chapter 8, "Command Reference" for the correct options.

SP2-0552 Bind variable *variable_name* not declared

Cause: The specified bind variable was not declared.

Action: Check the search string to make sure that it is valid.

SP2-0556 Invalid file name

Cause: Missing file name or an invalid file name specified.

Action: Make sure that a file name was specified.

SP2-0559 EXEC[UTE] statement

Cause: Usage message of the EXECUTE command.

Action: Check the syntax of the EXECUTE command in Chapter 8, "Command Reference" for the correct usage.

SP2-0560 Usage: DESCRIBE [*schema*.]*object*[.*subobject*|*@db_link*] [*column*]

Cause: Usage message of the DESCRIBE command.

Action: Check the syntax of the DESCRIBE command in Chapter 8, "Command Reference" for the correct usage.

SP2-0561 Object does not exist

Cause: The specified object you tried to DESCRIBE does not exist in the database.

Action: Retry the command with a valid object name.

SP2-0562 Object does not exist in package

Cause: The specified object you tried to DESCRIBE does not exist in the package.

Action: Check and make sure that the object name is correct.

SP2-0564 Object *object_name* is INVALID, it may not be described

Cause: The specified object you tried to DESCRIBE is invalid.

Action: Re-validate the object.

SP2-0565 Illegal identifier

Cause: Invalid character used in the DESCRIBE command.

Action: Correct the character and try again.

SP2-0566 Illegal sub-object specification

Cause: Invalid sub-object specification in the DESCRIBE command.

Action: Correct the subject specification and try again.

SP2-0567 Illegal column specification for PL/SQL object

Cause: It is illegal to describe a column within an object in the DESCRIBE command.

Action: Remove the column specification in the DESCRIBE command and try again.

SP2-0568 No bind variables declared

Cause: There are no bind variables declared.

Action: No action required.

SP2-0570 Usage: SET SERVEROUTPUT [ON | OFF] [SIZE [SIZE n].[FOR[MAT] [WRA[PPED] | WOR[D_WRAPPED] | TRU[NCATED]]]

Cause: An invalid option was used in the SET SERVEROUTPUT command.

Action: Specify a valid option.

SP2-0577 Usage: SET FLAGGER [OFF | ENTRY | INTERMEDIATE | FULL]

Cause: An invalid option was specified in the SET FLAGGER command.

Action: Specify a valid option.

SP2-0581 Object *object_name* is a package; use 'DESCRIBE *package.procedure*'

Cause: A package cannot be described as a stand-alone, it must be supplied with a sub-object, such as procedure.

Action: Use the DESCRIBE command to describe a sub-object within a package.

SP2-0582 Usage: [EXIT | QUIT] [SUCCESS | FAILURE | WARNING | n |<*variable*> | :<*bindvariable*>] [COMMIT | ROLLBACK]

Cause: An option to EXIT was invalid in SQL*Plus.

Action: Specify a valid option.

SP2-0584 EXIT variable *variable_name* was non-numeric

Cause: The specified EXIT variable is non-numeric.

Action: Check the syntax of the EXIT command in Chapter 8, "Command Reference" for the correct usage.

SP2-0590 A COMPUTE function must appear before each LABEL keyword

Cause: The function COMPUTE must appear before each LABEL keyword.

Action: Check the syntax of the COMPUTE command in Chapter 8, "Command Reference" for the correct usage.

SP2-0591 Unable to allocate dynamic space needed (*number_of_bytes* bytes) Try reducing ARRAYSIZE or the number of columns selected

Cause: Unable to allocate memory to process the command.

Action: Free up additional memory by: closing applications not required; reducing the size of the command, or statement; or by recoding the query to select fewer records.

SP2-0593 Label text must follow the LABEL keyword

Cause: Missing label text about the LABEL keyword in the COMPUTE command.

Action: Check the syntax of the COMPUTE command in Chapter 8, "Command Reference" for the correct options.

SP2-0594 Usage: SET COLSEP [" \" *text*]

Cause: Usage for SET COLSEP command.

Action: Specify a valid option.

SP2-0596 Usage: SET AUTO[COMMIT] [OFF | ON | IMM[EDIATE]]

Cause: An invalid option was used in the SET AUTO[COMMIT].

Action: Check the syntax of the SET AUTOCOMMIT command in Chapter 8, "Command Reference" for the correct options.

SP2-0597 *datatype _name* is not a valid *datatype _name* format

Cause: The value entered in the ACCEPT command was not in the specified datatype.

Action: Correct the datatype and re-enter.

SP2-0598 *value_name* does not match input format "*format_name*"

Cause: The value entered in the ACCEPT command was not in the specified format.

Action: Correct the format and try again.

SP2-0599 Usage: SET EDITF[ILE] *filename*[.*ext*]

Cause: Required filename was missing after the SET EDITFILE command.

Action: Check the syntax of the SET EDITFILE command in Chapter 8, "Command Reference" for the correct options.

SP2-0603 Usage: Illegal STORE command.
STORE [SET] *filename*[.*ext*] [CRE[ATE] | REP[LACE] | APP[END]]

Cause: An invalid STORE option was specified. Valid command clauses are CREATE, REPLACE or APPEND.

Action: Check the syntax of the STORE command in Chapter 8, "Command Reference" for the correct options.

SP2-0605 File *file_name* already exists. Use another name or STORE [SET] *filename*[.*ext*] REPLACE

Cause: The file specified in the STORE command already exists.

Action: Use the REPLACE option to overwrite the existing file, otherwise, specify another file name.

SP2-0606 Cannot create file *file_name*

Cause: The STORE command was unable to create the specified file. There may be insufficient disk space, too many open files, or read-only protection on the output directory.

Action: Check that there is enough disk space and that the protection on the directory allows creating a file.

SP2-0607 Cannot close file *file_name*

Cause: The STORE command was unable to close the specified file. Another resource may have locked the file.

Action: Check that the file is not locked before closing it.

SP2-0608 Object *object_name* is a remote object, cannot further describe

Cause: Unable to DESCRIBE the remote object.

Action: No action required.

SP2-0609 Usage: SET AUTOT[RACE] [OFF | ON | TRACE[ONLY]] [EXP[LAIN]] [STAT[ISTICS]]

Cause: An invalid option was used in the SET AUTOTRACE command.

Action: Check the syntax of the SET AUTOTRACE command in Chapter 8, "Command Reference" for the correct options.

SP2-0610 Error initializing *feature_name*

Cause: Not enough memory to enable this feature.

Action: Free up additional memory by closing applications not required, or reduce the size of the command, statement or query output.

SP2-0612 Error generating *report_name* report

Cause: Unable to generate the report using AUTOTRACE.

Action: Make a note of the message and the number, then contact the Database Administrator.

SP2-0613 Unable to verify PLAN_TABLE format or existence

Cause: An AUTOTRACE command was issued when the current user did not have the appropriate privilege to execute it.

Action: Make sure AUTOTRACE has the privileges and the objects to run. Make a note of the message and the number, then contact the Database Administrator.

SP2-0614 Server version too low for this feature

Cause: The current version of the Oracle Server is too low for this feature.

Action: Use a higher version of the Oracle Server.

SP2-0617 Cannot construct a unique STATEMENT_ID

Cause: Unable to construct a unique statement ID in AUTOTRACE.

Action: Check that AUTOTRACE is configured and that you have the PLUSTRACE role enabled.

SP2-0618 Cannot find the Session Identifier. Check PLUSTRACE role is enabled

Cause: Unable to find the session identifier.

Action: Check and make sure that the PLUSTRACE role is enabled.

SP2-0619 Error while connecting

Cause: An error occurred while AUTOTRACE attempted to make a second connection to the database instance.

Action: Check that the database limit on number of active sessions has not been exceeded.

SP2-0620 Error while disconnecting

Cause: An error occurred while AUTOTRACE attempted to disconnect from the database instance.

Action: Check that the database is still available.

SP2-0621 Error ORA *-error_number* while gathering statistics

Cause: No data was found in the PLAN_TABLE while gathering statistics while using AUTOTRACE.

Action: Refer to the *Oracle9i Error Messages* for the specified ORA error message.

SP2-0622 Starting line number must be less than ending line number

Cause: The starting line number specified is larger than the ending number.

Action: Re-enter the starting line number with a smaller line number.

SP2-0623 Error accessing PRODUCT_USER_PROFILE.
Warning: Product user profile information not loaded!
Error in disabling roles in product user profile

Cause: These error messages are warnings that the PRODUCT_USER_ PROFILE table has not been built in the SYSTEM account.

Action: The exact format of the file extension and location of the file are system dependent. See the SQL*Plus installation guide provided for your operating system. The script must be run as user SYSTEM.

SP2-0625 Error printing variable *variable_name*

Cause: Error encountered while printing the specified variable.

Action: Check and make sure that the specified variable is correct and try again.

SP2-0626 Error accessing package DBMS_APPLICATION_INFO

Cause: This message is followed by a successful login to the Oracle Server. The DBMS_APPLICATION package is used to maintain on-line information about a particular application logged onto Oracle. SET APPINFO could not be initialized.

Action: This package is created during the running of the CATPROC.SQL and should be available on all databases from Oracle 7.2. Check that your database is correctly installed.

SP2-0631 String beginning *string_name* is too long.
Maximum size is *string_length* characters

Cause: The string specified was too long.

Action: Reduce the size of the specified string and re-try the operation.

SP2-0640 Not connected.
PASSW[ORD] [*username*]

Cause: The PASSWORD command was issued when there was no connection to the Oracle instance

Action: Connect to the Oracle instance and re-try the operation; connect to the database before re-issuing the PASSWORD command.

SP2-0641 *command_name* requires connection to server

Cause: SQL*Plus was unable to execute the command because there was no connection to a database.

Action: Connect to a database and re-try the operation.

SP2-0642 SQL*Plus internal error state *error_state* context *error_number*. Unsafe to proceed

Cause: An internal error occurred.

Action: Make a note of the message and the numbers, then contact Oracle Support Services.

SP2-0645 Operating System error occurred
Unable to complete EDIT command

Cause: An operating system error occurred with the EDIT command.

Action: Check that the file was created successfully, and verify that the device you are writing to is still available.

SP2-0650 New passwords do not match

Cause: The new passwords entered did not match.

Action: Re-issue the PASSWORD command and make sure the same new passwords are entered correctly.

SP2-0659 Password unchanged

Cause: The PASSWORD command failed to change passwords because:

- No passwords were given.
- The new passwords did not match.

Action: Re-issue the PASSWORD command and make sure that the new passwords are entered correctly.

SP2-0666 WARNING: SHIFTINOUT only effects shift sensitive character sets

Cause: The NLS character set used in this session does not contain shift sensitive characters. The SET SHIFTINOUT command is unnecessary.

Action: No action required.

SP2-0667 Message file *facility<lang>.msb* not found

Cause: The SP1, SP2, or CPY message file could not be found. SQL*Plus cannot run.

Action: Check the Oracle platform specific documentation to make sure SQL*Plus is installed correctly. This may occur because the ORACLE_HOME environment variable or registry equivalent is not set to the location of the Oracle software. Make sure this value is set correctly. Check that the SQL*Plus binary message files exists in the SQL*Plus message directory, for example

$ORACLE_HOME/sqplus/mesg. Check the value of NLS_LANG environment variable or registry equivalent is correct.

SP2-0668 Invalid variable name

Cause: An invalid character was specified as part of the name.

Action: Specify the variable with valid characters.

SP2-0669 Valid characters are alphanumerics and '_'

Cause: An invalid character was specified as part of the name.

Action: Specify the variable with alphanumerics and '_'.

SP2-0670 Internal number conversion failed

Cause: A conversion request could not be performed because the string contained alphanumeric characters.

Action: Make sure that the string only contain numeric digits.

SP2-0675 COPY command not available

Cause: The COPY command is not available in this version of SQL*Plus.

Action: Make a note of the message and the number, then contact Oracle Support Services.

SP2-0676 Bind variable length cannot exceed *variable_length units_of_variable*

Cause: The length of the bind variable datatype was exceeded.

Action: Reduce the length of the bind variable datatype.

SP2-0678 Column or attribute type can not be displayed by SQL*Plus

Cause: The type specified is not supported.

Action: Rewrite the query to select the data with types that SQL*Plus supports.

SP2-0685 The date *entered_variable* is invalid or format mismatched *format*

Cause: An invalid date was entered or does not match the format.

Action: Enter a valid date or date format.

SP2-0686 Usage: DESCRIBE [*schema*.]*object*[*@db_link*]

Cause: Usage for the DESCRIBE command.

Action: Check the syntax of the DESCRIBE command in Chapter 8, "Command Reference" for the correct options.

SP2-0691 Expected SYSDBA or SYSOPER, not *command_name*

Cause: Attempted to use the CONNECT AS syntax and specified something other than SYSDBA or SYSOPER.

Action: Correct the syntax and issue the CONNECT command again.

SP2-0692 Usage: CONN[ECT] [*login*] [AS [SYSDBA | SYSOPER]]
Where <*login*> ::= <*username*>[/<*password*>][@<*connect_string*>] | /

Cause: Usage for SQL*Plus CONNECT command.

Action: Check the syntax for the CONNECT command in Chapter 8, "Command Reference" for the correct usage.

SP2-0714 Invalid combination of STARTUP options

Cause: The specified options of the STARTUP command cannot be used simultaneously.

Action: Check the syntax of the STARTUP command in Chapter 8, "Command Reference" for the correct usage.

SP2-0715 Invalid combination of SHUTDOWN options

Cause: The specified options of the SHUTDOWN command cannot be used simultaneously.

Action: Check the syntax of the SHUTDOWN command in Chapter 8, "Command Reference" for the correct usage.

SP2-0716 Invalid combination of ARCHIVE LOG options

Cause: The specified options of the ARCHIVE LOG command cannot be used simultaneously.

Action: Check the syntax of the ARCHIVE LOG command in Chapter 8, "Command Reference" for the correct usage.

SP2-0717 Illegal SHUTDOWN option

Cause: An invalid option was used in the SHUTDOWN command.

Action: Check the syntax of the SHUTDOWN command in Chapter 8, "Command Reference" for the correct options.

SP2-0718 Illegal ARCHIVE LOG option

Cause: An invalid option was used in the ARCHIVE LOG command.

Action: Check the syntax of the ARCHIVE LOG command in Chapter 8, "Command Reference" for the correct options.

SP2-0728 Specify log: [<RET>=suggested | *filename* | AUTO | CANCEL]

Cause: This is a RECOVER DATABASE command prompt, prompting for the redo log files to be applied.

Action: Enter one of the above options.

SP2-0729 Cannot SET INSTANCE while connected to a database

Cause: There is a problem with the connection instance while issuing the SET INSTANCE command.

Action: Disconnect from the instance before re-issuing the command.

SP2-0733 Invalid connect string

Cause: Invalid connect string was specified.

Action: Check and make sure that connect string is correct.

SP2-0734 Unknown command beginning *command_name* ... - rest of line ignored

Cause: The command entered was invalid.

Action: Check syntax and re-enter.

SP2-0735 Unknown *command_name* option beginning *option_name*

Cause: An invalid option was specified for a given command.

Action: Check the syntax in Chapter 8, "Command Reference" for the correct options of the command you used.

SP2-0736 Command line overflow while substituting into line beginning *string_name*

Cause: The maximum length of the command line was exceeded.

Action: Reduce the length of the data in the substitution variables used in the command.

SP2-0737 Usage: SET DESCRIBE [DEPTH [1 | n | ALL]] [LINENUM [ON | OFF]] [INDENT [ON | OFF]]

Cause: Usage message for SET DESCRIBE command.

Action: Check the syntax of the SET DESCRIBE command in Chapter 8, "Command Reference" for the correct options.

SP2-0738 Restricted command *command_name* not available

Cause: The command was restricted by the -RESTRICT command-line option for security reasons.

Action: Ask your systems administrator why SQL*Plus should be run with a "-RESTRICT" option.

SP2-0745 Usage: SET SQLPLUSCOMPAT[IBILITY] *version.release*.[*update*]

Cause: An invalid option was used in the SET SQLPLUSCOMPAT[IBLITY] command.

Action: Check the syntax of the SET SQLPLUSCOMPATIBLITY command in Chapter 8, "Command Reference" for the correct options.

SP2-0746 *command_option* option out of range (*lower* through *upper*)

Cause: The specified value was not in the range.

Action: Specify a value in the range.

SP2-0747 PAGESIZE must be at least *max_page_size* to run this query with LINESIZE *line_size*

Cause: The PAGESIZE setting was too small to display the specified LINESIZE.

Action: Increase the PAGESIZE to at least match the specified LINESIZE.

COPY Command Messages

CPY0002 Illegal or missing APPEND, CREATE, INSERT, or REPLACE option

Cause: An internal COPY function has invoked COPY with a create option (flag) value that is out of range.

Action: Please contact Oracle Worldwide Customer Support Services.

CPY0003 Internal Error: logical host number out of range

Cause: An internal COPY function has been invoked with a logical host number value that is out of range.

Action: Please contact Oracle Worldwide Customer Support Services.

CPY0004 Source and destination table and column names don't match

Cause: On an APPEND operation or an INSERT (when the table exists), at least one column name in the destination table does not match the corresponding column name in the optional column name list or in the SELECT command.

Action: Re-specify the COPY command, making sure that the column names and their respective order in the destination table match the column names and column order in the optional column list or in the SELECT command

CPY0005 Source and destination column attributes don't match

Cause: On an APPEND operation or an INSERT (when the table exists), at least one column in the destination table does not have the same datatype as the corresponding column in the SELECT command.

Action: Re-specify the COPY command, making sure that the data types for items being selected agree with the destination. Use TO_DATE, TO_CHAR, and TO_NUMBER to make conversions.

CPY0006 Select list has more columns than destination table

Cause: On an APPEND operation or an INSERT (when the table exists), the number of columns in the SELECT command is greater than the number of columns in the destination table.

Action: Re-specify the COPY command, making sure that the number of columns being selected agrees with the number in the destination table.

CPY0007 Select list has fewer columns than destination table

Cause: On an APPEND operation or INSERT (when the table exists), the number of columns in the SELECT command is less than the number of columns in the destination table.

Action: Re-specify the COPY command, making sure that the number of columns being selected agrees with the number in the destination table.

CPY0008 More column list name than columns in the destination table

Cause: On an APPEND operation or an INSERT (when the table exists), the number of columns in the column name list is greater than the number of columns in the destination table.

Action: Re-specify the COPY command, making sure that the number of columns in the column list agrees with the number in the destination table.

CPY0009 Fewer column list name than columns in the destination table

Cause: On an APPEND operation or an INSERT (when the table exists), the number of columns in the column name list is less than the number of columns in the destination table.

Action: Re-specify the COPY command, making sure that the number of columns in the column list agrees with the number in the destination table.

B

Release 9.0.1 Enhancements

This appendix describes the enhancements for SQL*Plus Release 9.0.1.

- SQLPLUS Command Line Switches
- iSQL*Plus
- START, @, @@
- SET APPINFO
- SET SQLPLUSCOMPATIBILITY
- HR Sample Schema

SQL*Plus Release 9.0.1 Enhancements

SQLPLUS Command Line Switches

The version and usage command line switches have been standardized as:

```
-V[ERSION]
-H[HELP]
```

Invalid options give the usage message corresponding to -HELP. The "-" and "-?" options have been obsoleted.

*i*SQL*Plus

*i*SQL*Plus is a browser-based interface to SQL*Plus. *i*SQL*Plus is only available on the Microsoft Windows operating system in this release. For more information on *i*SQL*Plus, see the *iSQL*Plus User's Guide and Reference,* A88826-01.

START, @, @@

New commands were added to read and execute a SQL script from a Uniform Resource Identifier (URI). The ability to read and execute a script from a URI is available in the @, @@ and START commands. The syntax is:

```
@{uri|file_name[.ext]} [arg...]
@@file_name[.ext]
STA[RT] {uri|file_name[.ext]} [arg...]
```

"uri" specifies a script to run on the specified web server. SQL*Plus supports HTTP, FTP and gopher protocols. Pass variable values to the script in the usual way, for example:

```
http://host.domain/script.sql value1 value2
```

Note: This feature is only available on Microsoft Windows.

SET APPINFO

The default for SET APPINFO is now OFF. This allows some privileged DBA operations to be executed when the database is not fully configured. Add SET APPINFO ON to *glogin.sql* to get the previous behavior.

SET SQLPLUSCOMPATIBILITY

There is a new SET command, SET SQLPLUSCOMPAT[IBILITY] {x.y[.z]}. It sets the behavior of the VARIABLE command to that of the version specified by SQLPLUSCOMPATIBILITY.

In later releases of SQL*Plus, SQLPLUSCOMPATIBILITY may support features other than VARIABLE. The SQLPLUSCOMPATIBILITY command can be used to help keep the expected behavior of a release.

HR Sample Schema

The SQL*Plus documentation uses a new set of sample schemas for all examples. See the *Oracle9i Sample Schemas* guide for details on access, implementation and use of the sample schemas.

C

SQL*Plus Limits

Table C-1, on the following page, lists the limit, or maximum value, of each of the SQL*Plus elements shown. The limits shown are valid for most operating systems.

Table 0–1 SQL*Plus Limits

Item	Limit
filename length	system dependent
username length	30 bytes
user variable name length	30 bytes
user variable value length	240 characters
command-line length	2500 characters
length of a LONG value entered through SQL*Plus	LINESIZE value
LINESIZE	system dependent
LONGCHUNKSIZE value	system dependent
output line size	system dependent
line size after variable substitution	3,000 characters (internal only)
number of characters in a COMPUTE command label	500 characters
number of lines per SQL command	500 (assuming 80 characters per line)
maximum PAGESIZE	50,000 lines
total row width	60,000 characters for VMS; otherwise, 32,767 characters
maximum ARRAYSIZE	5000 rows
maximum number of nested command files	20 for VMS, CMS, Unix; otherwise, 5
maximum page number	99,999
maximum PL/SQL error message size	2K
maximum ACCEPT character string length	240 Bytes
maximum number of DEFINE variables	2048

D

SQL Command List

Table D-1, on the following page, lists major SQL commands. Refer to the *Oracle9i SQL Reference* for full documentation of these commands.

Table 0–1 SQL Command List

Major SQL Commands and Clauses	
ALTER	LOCK TABLE
ANALYZE	NOAUDIT
AUDIT	RENAME
COMMENT	REVOKE
CREATE	SAVEPOINT
DROP	SET ROLE
EXPLAIN	SET TRANSACTION
GRANT	TRUNCATE
INSERT	UPDATE

E

Security

This appendix describes the available methods for controlling access to database tables and SQL*Plus commands. It covers the following topics:

- PRODUCT_USER_PROFILE Table
- Disabling SQL*Plus, SQL, and PL/SQL Commands
- Creating and Controlling Roles
- Disabling Commands with SQLPLUS -RESTRICT

PRODUCT_USER_PROFILE Table

Various Oracle products use the PRODUCT_USER_PROFILE (PUP) table, a table in the SYSTEM account, to provide product-level security that supplements the user-level security provided by the SQL GRANT and REVOKE commands and user roles.

DBAs can use the PUP table to disable certain SQL and SQL*Plus commands in the SQL*Plus environment on a per-user basis. SQL*Plus—not Oracle—enforces this security. DBAs can even restrict access to the GRANT, REVOKE, and SET ROLE commands to control users' ability to change their database privileges.

SQL*Plus reads restrictions from the PUP table when a user logs in to SQL*Plus and maintains those restrictions for the duration of the session. Changes to the PUP table will only take effect the next time the affected users log in to SQL*Plus.

When SYSTEM, SYS, or a user authenticating with AS SYSDBA or AS SYSOPER privileges connects or logs in, SQL*Plus does not read the PUP table. Therefore, no restrictions apply to these users.

The PUP table applies only to the local database. If accessing objects on a remote database via a database link, the PUP table for the remote database does not apply. The remote database cannot extract the username and password from the database link in order to determine that user's profile and privileges.

Creating the PUP Table

You can create the PUP table by running the command file named PUPBLD with the extension SQL as SYSTEM. The exact format of the file extension and the location of the file are system dependent. See the Oracle installation and user's manual(s) provided for your operating system or your DBA for more information.

Note: If the table is created incorrectly, all users other than privileged users will see a warning when connecting to Oracle that the PUP table information is not loaded.

PUP Table Structure

The PUP table has the following columns:

```
PRODUCT                 NOT NULL CHAR (30)
USERID                  CHAR(30)
ATTRIBUTE               CHAR(240)
SCOPE                   CHAR(240)
```

```
NUMERIC_VALUE           NUMBER(15,2)
CHAR_VALUE              CHAR(240)
DATE_VALUE              DATE
LONG_VALUE              LONG
```

Description and Use of PUP Columns

Refer to the following list for the descriptions and use of each column in the PUP table:

Product — Must contain the product name (in this case "SQL*PLUS"). You cannot enter wildcards or NULL in this column.

Userid — Must contain the username (in uppercase) of the user for whom you wish to disable the command. To disable the command for more than one user, use SQL wild cards (%) or make multiple entries. Thus, all of the following entries are valid:

- HR
- CLASS1
- CLASS% (all users whose names start with CLASS)
- % (all users)

Attribute — Must contain the name (in uppercase) of the SQL, SQL*Plus, or PL/SQL command you wish to disable (for example, GET). If you are disabling a role, it must contain the character string “ROLES”. You cannot enter a wildcard. See the section “Administration” later in this chapter for a list of SQL and SQL*Plus commands you can disable. See the section "Disabling Commands with SQLPLUS -RESTRICT" in this chapter for information on how to disable a role.

Scope — SQL*Plus ignores this column. It is recommended that you enter NULL in this column. Other products may store specific file restrictions or other data in this column.

Numeric_Value — SQL*Plus ignores this column. It is recommended that you enter NULL in this column. Other products may store numeric values in this column.

Char_Value	Must contain the character string “DISABLED” to disable a SQL, SQL*Plus, or PL/SQL command. If you are disabling a role, it must contain the name of the role you wish to disable. You cannot use a wildcard. See "Disabling Commands with SQLPLUS -RESTRICT" below for information on how to disable a role.
Date_Value	SQL*Plus ignores this column. It is recommended that you enter NULL in this column. Other products may store DATE values in this column.
Long_Value	SQL*Plus ignores this column. It is recommended that you enter NULL in this column. Other products may store LONG values in this column.

PUP Table Administration

The DBA username SYSTEM owns and has all privileges on the PUP table. Other Oracle usernames should have only SELECT access to this table, which allows a view of restrictions of that username and those restrictions assigned to PUBLIC. The command file PUPBLD.SQL, when run, grants SELECT access on the PUP table to PUBLIC.

Disabling SQL*Plus, SQL, and PL/SQL Commands

To disable a SQL or SQL*Plus command for a given user, insert a row containing the user’s username in the Userid column, the command name in the Attribute column, and DISABLED in the Char_Value column.

The Scope, Numeric_Value, and Date_Value columns should contain NULL. For example:

```
PRODUCT    USERID  ATTRIBUTE  SCOPE   NUMBERIC     CHAR       DATE
                                      VALUE        VALUE      VALUE
-------    ------  ---------  -----   --------     ------     -----
SQL*Plus   HR      HOST                            DISABLED
SQL*Plus   %       INSERT                          DISABLED
SQL*Plus   %       UPDATE                          DISABLED
SQL*Plus   %       DELETE                          DISABLED
```

To reenable commands, delete the row containing the restriction.

You can disable the following SQL*Plus commands:

COPY	PASSWORD
EDIT	QUIT
EXECUTE	RUN
EXIT	SAVE
GET	SET (see note below)
HOST (or your operating system's alias for HOST, such as $ on VMS, and ! on UNIX)	SPOOL
	START

Note: Disabling the SQL*Plus SET command will also disable the SQL SET ROLE and SET TRANSACTION commands. Disabling the SQL*Plus START command will also disable the SQL*Plus @ and @@ commands.

You can also disable the following SQL commands:

ALTER	LOCK
ANALYZE	NOAUDIT
AUDIT	RENAME
CONNECT	REVOKE
CREATE	SELECT
DELETE	SET ROLE
DROP	SET TRANSACTION
GRANT	TRUNCATE
INSERT	UPDATE

You can also disable the following PL/SQL commands:

BEGIN	DECLARE

Note: Disabling BEGIN and DECLARE does not prevent the use of the SQL*Plus EXECUTE command. EXECUTE must be disabled separately.

Example E–1 Setting Restrictions in the PUP Table

This is an example of how to insert a row into the PUP table to restrict the user HR from using the SELECT statement:

1. Log in as SYSTEM with the command

```
SQLPLUS SYSTEM/MANAGER
```

2. Insert a row into the PUP table with the command:

```
INSERT INTO PRODUCT_USER_PROFILE
VALUES ('SQL*PLUS', 'HR', 'SELECT', NULL, NULL, 'DISABLED', NULL, NULL);
```

3. Connect as HR/HR and try to SELECT something:

```
CONNECT HR/HR;
SELECT * FROM EMP_DETAILS_VIEW;
```

This command causes the following error message:

```
SP2-0544: INVALID COMMAND: SELECT
```

4. To delete this row and remove the restriction from the user HR, CONNECT again as SYSTEM and enter:

```
DELETE FROM PRODUCT_USER_PROFILE WHERE USERID = 'HR';
```

Creating and Controlling Roles

You can use SQL commands to create and control access to roles to provide security for your database tables.

By creating a role and then controlling who has access to it, you can ensure that only certain users have access to particular database privileges.

Roles are created and used with the SQL CREATE, GRANT, and SET commands:

- To create a role, you use the CREATE command. You can create roles with or without passwords.

- To grant access to roles, you use the GRANT command. In this way, you can control who has access to the privileges associated with the role.
- To access roles, you use the SET ROLE command. If you created the role with a password, the user must know the password in order to access the role.

For more information about roles, see your *Oracle9i SQL Reference*, your *Oracle9i Administrator's Guide*, and your *Oracle9i Concepts* manual.

Disabling SET ROLE

From SQL*Plus, users can submit any SQL command. In certain situations, this can cause security problems. Unless you take proper precautions, a user could use SET ROLE to access privileges obtained via an application role. With these privileges, they might issue SQL statements from SQL*Plus that could wrongly change database tables.

To prevent application users from accessing application roles in SQL*Plus, you can use the PUP table to disable the SET ROLE command. You also need to disable the BEGIN and SQL*Plus EXECUTE commands to prevent application users setting application roles through a PL/SQL block. This allows a SQL*Plus user only those privileges associated with the roles enabled when they started SQL*Plus. For more information about the creation and usage of user roles, see your *Oracle9i SQL Reference* and *Oracle9i Administrator's Guide*.

Disabling User Roles

To disable a role for a given user, insert a row in the PUP table containing the user's username in the Userid column, "ROLES" in the Attribute column, and the role name in the Char_Value column.

Note: When you enter "PUBLIC" or "%" for the Userid column, you disable the role for all users. You should only use "%" or "PUBLIC" for roles which are granted to "PUBLIC". If you try to disable a role that has not been granted to a user, none of the roles for that user are disabled.

The Scope, Numeric_Value, and Date_Value columns should contain NULL. For example:

```
PRODUCT    USERID  ATTRIBUTE  SCOPE   NUMERIC      CHAR       DATE
                                      VALUE        VALUE      VALUE
-------    ------  ---------  -----   --------     ------     -----
SQL*Plus   HR      ROLES                           ROLE1
SQL*Plus   PUBLIC  ROLES                           ROLE2
```

During login, these table rows are translated into the command

```
SET ROLE ALL EXCEPT ROLE1, ROLE2
```

To ensure that the user does not use the SET ROLE command to change their roles after login, you can disable the SET ROLE command. See "Disabling SET ROLE" earlier in this appendix.

To reenable roles, delete the row containing the restriction.

Disabling Commands with SQLPLUS -RESTRICT

Like the Product User Profile table, the RESTRICT option allows you to disable certain commands that interact with the operating system. However, commands disabled with the -RESTRICT option are disabled even when no connection to a server exists, and remain disabled until SQL*Plus terminates.

The following table shows which commands are disabled in each restriction level.

Command	Level 1	Level 2	Level 3
EDIT	disabled	disabled	disabled
GET			disabled
HOST, !	disabled	disabled	disabled
SAVE		disabled	disabled
SPOOL		disabled	disabled
START, @, @@			disabled
STORE		disabled	disabled

For more information about the RESTRICT option, see the SQLPLUS -R[ESTRICT] {1 | 2 | 3} command on page 7-7

F

Obsolete SQL*Plus Commands

This appendix covers earlier versions of some SQL*Plus commands. While these older commands still function within SQL*Plus, they are no longer supported. It is recommended that you use the alternative SQL*Plus commands listed in the following table.

SQL*Plus Obsolete Command Alternatives

Obsolete Command	Alternative Command	Description of Alternative Command
BTITLE (old form)	BTITLE on page 8-23	Places and formats a title at the bottom of each report page or lists the current BTITLE definition.
COLUMN DEFAULT	COLUMN CLEAR on page 8-30	Resets column display attributes to default values.
DOCUMENT	REMARK on page 8-86	Places a comment which SQL*Plus does not interpret as a command.
NEWPAGE	SET NEWPAGE on page 8-106	Sets the number of blank lines to be printed from the top of each page to the top title.
SET BUFFER	EDIT on page 8-63	Enables the editing of the SQL*Plus command buffer, or the contents of a saved file. Use the SQL*Plus SAVE, GET, @ and START commands to create and use external files.
SET CLOSECURSOR	none	Obsolete
SET DOCUMENT	none	Obsolete
SET MAXDATA	none	Obsolete
SET SCAN	SET DEFINE on page 8-102	Sets the character used to prefix substitution variables.
SET SPACE	SET COLSEP on page 8-101	Sets the text to be printed between SELECTed columns.
SET TRUNCATE	SET WRAP on page 8-111	Controls whether SQL*Plus truncates a SELECTed row if it is too long for the current line width.
SHOW LABEL	none	Obsolete
TTITLE (old form)	TTITLE on page 8-138	Places and formats a title at the top of each report page or lists the current TTITLE definition.

BTITLE (old form)

Purpose

Displays a title at the bottom of each report page.

Syntax

BTI[TLE] *text*

Usage Notes

The old form of BTITLE offers formatting features more limited than those of the new form, but provides compatibility with UFI (a predecessor of SQL*Plus). The old form defines the bottom title as an empty line followed by a line with centered text. Refer to TTITLE (old form) in this appendix for more details.

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] {ON|OFF}

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] {ON|OFF}

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 now 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 {ON|OFF}

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] {ON|OFF}

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

TTITLE (old form)

Purpose

Displays a title at the top of each report page.

Syntax

TTI[TLE] *text*

Usage Notes

The old form of TTITLE offers formatting features more limited than those of the new form, but provides compatibility with UFI (a predecessor of SQL*Plus). The old form defines the top title as a line with the date left-aligned and the page number right-aligned, followed by a line with centered text and then a blank line.

The *text* you enter defines the title TTITLE will display.

SQL*Plus centers text based on the size of a line as determined by SET LINESIZE. A separator character (|) begins a new line; two line separator characters in a row (||) insert a blank line. You can change the line separator character with SET HEADSEP.

You can control the formatting of page numbers in the old forms of TTITLE and BTITLE by defining a variable named "_page". The default value of _page is the formatting string "page &P4". To alter the format, you can DEFINE _page with a new formatting string as follows:

```
SET ESCAPE / SQL> DEFINE _page = 'Page /&P2'
```

This formatting string will print the word "page" with an initial capital letter and format the page number to a width of two. You can substitute any text for "page" and any number for the width. You must set escape so that SQL*Plus does not interpret the ampersand (&) as a substitution variable. See the ESCAPE variable of the SET command in Chapter 8 for more information on setting the escape character.

SQL*Plus interprets TTITLE in the old form if a valid new-form clause does not immediately follow the command name.

If you want to use CENTER with TTITLE and put more than one word on a line, you should use the new form of TTITLE. For more information see the TTITLE command in Chapter 8.

Example

To use the old form of TTITLE to set a top title with a left-aligned date and right-aligned page number on one line followed by SALES DEPARTMENT on the next line and PERSONNEL REPORT on a third line, enter

```
TTITLE 'SALES DEPARTMENT|PERSONNEL REPORT'
```

Glossary

account

An authorized user of an operating system or a product (such as Oracle database server or Oracle Forms). Depending on the operating system, may be referred to as ID, User ID, login, and so on. Accounts are often created and controlled by a system administrator.

alias

In SQL, a temporary name assigned to a table, view, column, or value within a SQL statement, used to refer to that item later in the same statement or in associated SQL*Plus commands.

alignment

The way in which data is positioned in a field. It may be positioned to the left, right, center, flush / left, flush / right, or flush / center of the defined width of a field.

anonymous block

A PL / SQL program unit that has no name and does not require the explicit presence of the BEGIN and END keywords to enclose the executable statements.

archived redo log

Recovery structure where online redo log files are archived before being reused.

ARCHIVELOG

Redo log mode where the filled online redo log files are archived before they are reused in the cycle. In ARCHIVELOG mode, the database can be completely recovered from both instance and disk failure. The database can also be backed up while it is open and available for use. However, additional administrative

operations are required to maintain the archived redo log. See also archived redo log.

argument

A data item following the command file name in a START command. The argument supplies a value for a parameter in the command file.

array processing

Processing performed on multiple rows of data rather than one row at a time. In some Oracle utilities such as SQL*Plus, Export/Import, and the precompilers, users can set the size of the array; increasing the array size often improves performance.

ASCII

A convention for using digital data to represent printable characters. ASCII is an acronym for American Standard Code for Information Interchange.

autocommit

A feature unique to SQL*Plus that enables SQL*Plus to automatically commit changes to the database after every successful execution of a SQL command or PL/SQL block. Setting the AUTOCOMMIT variable of the SET command to ON enables this feature. Setting the AUTOCOMMIT variable to *n* enables this feature after every *n* successful INSERT, UPDATE or DELETE commands or PL/SQL blocks.

background process

A non-interactive process that runs in an operating system environment and performs some service or action. Certain Oracle database server products use background processes for different tasks, such as performing and coordinating tasks on behalf of concurrent users of the database, processing and delivering electronic messages, and managing printing services.

bind reference

A reference to a parameter used to replace a single literal value (for example, a character string, number, or date) appearing anywhere in a PL/SQL construct or a SQL SELECT statement. For a bind reference, you must precede the parameter name with a colon (:).

bind variable

A variable in a SQL statement that must be replaced with a valid value, or the address of a value, in order for the statement to successfully execute.

bit

The smallest unit of data. A bit only has two possible values, 0 or 1. Bits can be combined into groups of eight called bytes; each byte represents a single character of data. See also byte.

block

In PL/SQL, a group of SQL and PL/SQL commands related to each other through procedural logic.

body

A report region that contains the bulk of the report (text, graphics, data, and computations).

break

An event, such as a change in the value of an expression, that occurs while SQL*Plus processes a query or report. You can direct SQL*Plus to perform various operations, such as printing subtotals, whenever specified breaks occur.

break column

A column in a report that causes a break when its value changes and for which the user has defined break operations.

break group

A group containing one or more break columns.

break hierarchy

The order in which SQL*Plus checks for the occurrence of breaks and triggers the corresponding break operations.

break order

Indicates the order in which to display a break column's data. Valid options are Ascending and Descending.

break report

A report that divides rows of a table into "sets", based on a common value in the break column.

buffer

An area where the user's SQL statements or PL/SQL blocks are temporarily stored. The SQL buffer is the default buffer. You can edit or execute commands from multiple buffers; however, SQL*Plus does not require the use of multiple buffers.

byte

A group of eight sequential bits that represents a letter, number, or symbol (that is, a character). Treated as a unit of data by a computer.

CGI script

See Common Gateway Interface.

CHAR datatype

An Oracle datatype provided for ANSI/ISO compatibility. A CHAR column is a fixed-length column and can contain any printable characters, such as A, 3, &, or blanks, and can have from 1 to 2000 bytes or can be null. For more information about the CHAR datatype, refer to the *Oracle9i SQL Reference*.

character

A single location on a computer system capable of holding one alphabetic character or numeric digit. One or more characters are held in a field. One or more fields make up a record, and one or more records may be held in a file or database table.

character string

A group of sequential letters, numerals, or symbols, usually comprising a word or name, or portion thereof.

clause

A part of a SQL statement that does not constitute the full statement; for example, a "WHERE clause".

client

A user, software application, or computer that requests the services, data, or processing of another application or computer (the "server"). In a two-task environment, the client is the user process. In a network environment, the client is the local user process and the server may be local or remote.

CLOB datatype

A standard Oracle datatype. The CLOB datatype is used to store single-byte character large object data, and can store up to 4 gigabytes of character data.

column

A vertical space in a database table that represents a particular domain of data. A column has a column name and a specific datatype. For example, in a table of employee information, all of the employees' dates of hire would constitute one column. A record group column represents a database column.

column expression

An expression in a SELECT statement that defines which database column(s) are retrieved. It may be a column name or a valid SQL expression referencing a column name.

column heading

A heading created for each column appearing in a report.

command

An instruction to or request of a program, application, operating system, or other software, to perform a particular task. Commands may be single words or may require additional phrases, variously called arguments, options, parameters, and qualifiers. Unlike statements, commands execute as soon as you enter them. ACCEPT, CLEAR, and COPY are examples of commands in SQL*Plus.

fcommand file

A file containing a sequence of commands that you can otherwise enter interactively. The file is saved for convenience and re-execution. Command files are often called by operating-system specific names. In SQL*Plus, you can execute the command file with the START, @ or @@ commands.

command line

A line on a computer display on which typed in commands appear. An example of a command line is the area next to the DOS prompt on a personal computer. See also prompt.

command prompt

The text, by default SQL>, with which SQL*Plus requests your next command.

comment

A language construct for the inclusion of explanatory text in a program, the execution of which remains unaffected.

commit

To make permanent changes to data (inserts, updates, deletes) in the database. Before changes are committed, both the old and new data exist so that changes can be stored or the data can be restored to its prior state.

Common Gateway Interface

The Common Gateway Interface (CGI) describes a part of a web server that allows user interaction, typically via a web browser, with programs running on the server. CGI scripts enable this user interaction to create dynamic web pages or web page elements, or to take user input and respond accordingly. A very common use is to provide an interactive form which a user completes online and then submits. Some common languages in use for CGI scripts are Perl, JavaScript and Java

computation

Used to perform runtime calculations on data fetched from the database. These calculations are a superset of the kinds of calculations that can be done directly with a SELECT statement. See also formula column.

computed column

See computation.

configuration

In Oracle Net, the set of instructions for preparing network communications, as outlined in the Oracle Net documentation.

configuration files

Files that are used to identify and characterize the components of a network. Configuration is largely a process of naming network components and identifying relationships among those components.

connect

To identify yourself to Oracle by entering your username and password in order to gain access to the database. In SQL*Plus, the CONNECT command allows you to log off Oracle and then log back on with a specified username.

connect identifier

The set of parameters, including a protocol, that Oracle Net uses to connect to a specific Oracle instance on the network.

current line

In an editor, such as the SQL*Plus editor, the line in the current buffer that editing commands will currently affect.

database

A set of operating system files, treated as a unit, in which an Oracle database server stores a set of data dictionary tables and user tables. A database requires three types of files: database files, redo log files, and control files.

database administrator (DBA)

(1) A person responsible for the operation and maintenance of an Oracle database server or a database application. The database administrator monitors its use in order to customize it to meet the needs of the local community of users. (2) An Oracle username that has been given DBA privileges and can perform database administration functions. Usually the two meanings coincide. There may be more than one DBA per site.

database instance failure

Failure that occurs when a problem arises that prevents an Oracle database instance (SGA and background processes) from continuing to work. Instance failure may result from a hardware problem such as power outage, or a software problem, such as operating system crash. When an instance failure occurs, the data in the buffers of the SGA is not written to the datafiles.

database link

An object stored in the local database that identifies a remote database, a communication path to the remote database, and optionally, a username and password for it. Once defined, a database link can be used to perform queries on tables in the remote database. Also called *DBlink*. In SQL*Plus, you can reference a database link in a DESCRIBE or COPY command.

database object

Something created and stored in a database. Tables, views, synonyms, indexes, sequences, clusters, and columns are all examples of database objects.

database server

The computer which runs the ORACLE Server kernel and contains the database.

database specification

An alphanumeric code that identifies a database, used to specify the database in Oracle Net operations and to define a database link. In SQL*Plus, you can reference a database specification in a COPY, CONNECT, or SQLPLUS command.

database string

A string of Oracle Net parameters used to indicate the network prefix, the host system you want to connect to, and the system ID of the database on the host system.

Data Control Language (DCL)

The category of SQL statements that control access to the data and to the database. Examples are the GRANT and REVOKE statements. Occasionally DCL statements are grouped with DML statements.

Data Definition Language (DDL)

The category of SQL statements that define or delete database objects such as tables or views. Examples are the CREATE, ALTER, and DROP statements.

data dictionary

A comprehensive set of tables and views automatically created and updated by the Oracle database server, which contains administrative information about users, data storage, and privileges. It is installed when Oracle is initially installed and is a central source of information for the Oracle database server itself and for all users of Oracle. The tables are automatically maintained by Oracle. It is sometimes referred to as the *catalog*.

Data Manipulation Language (DML)

The category of SQL statements that query and update the database data. Common DML statements are SELECT, INSERT, UPDATE, and DELETE. Occasionally DCL statements are grouped with DML statements.

data security

The mechanisms that control the access and use of the database at the object level. For example, data security includes access to a specific schema object and the specific types of actions allowed for each user on the object (for example, user HR can issue SELECT and INSERT statements but not DELETE statements using the EMP table). It also includes the actions, if any, that are audited for each schema object.

datatype

(1) A standard form of data. The Oracle datatypes are CHAR, NCHAR, VARCHAR2, NVARCHAR2, DATE, NUMBER, LONG, CLOB, NCLOB, RAW, and LONG RAW; however, the Oracle database server recognizes and converts other standard datatypes. (2) A named set of fixed attributes that can be associated with an item as a property. Data typing provides a way to define the behavior of data.

DATE datatype

A standard Oracle datatype used to store date and time data. Standard date format is DD-MMM-YY, as in 23-NOV-98. A DATE column may contain a date and time between January 1, 4712 BC to December 31, 9999 AD.

DBA

See database administrator (DBA).

DCL

See Data Control Language (DCL).

DDL

See Data Definition Language (DDL).

default

A clause or option value that SQL*Plus uses if you do not specify an alternative.

default database

See local database.

directory

On some operating systems, a named storage space for a group of files. It is actually one file that lists a set of files on a particular device.

dismounted database

A database that is not mounted by any instance, and thus cannot be opened and is not currently available for use.

display format

See format.

display width

The number of characters or spaces allowed to display the values for an output field.

DML

See Data Manipulation Language (DML).

DUAL table

A standard Oracle database table named DUAL, which contains exactly one row. The DUAL table is useful for applications that require a small "dummy" table (the data is irrelevant) to guarantee a known result, such as "true."

editor

A program that creates or modifies files.

end user

The person for whom a system is being developed; for example, an airline reservations clerk is an end user of an airline reservations system. See also SQL*Plus.

error message

A message from a computer program (for example, SQL*Plus) informing you of a potential problem preventing program or command execution.

expression

A formula, such as SALARY + COMMISSION, used to calculate a new value from existing values. An expression can be made up of column names, functions, operators, and constants. Formulas are found in commands or SQL statements.

extension

On some operating systems, the second part of the full file specification. Several standard file extensions are used to indicate the type or purpose of the file, as in file extensions of SQL, LOG, LIS, EXE, BAT, and DIR. Called file type on some operating systems.

file

A collection of data treated as a unit, such as a list, document, index, note, set of procedures, and so on. Generally used to refer to data stored on magnetic tapes or disks. See also filename, extension, and file type.

filename

The name component of a file specification. A filename is assigned by either the user or the system when the file itself is created. See also extension and file type.

file type

On some operating systems, the part of the filename that usually denotes the use or purpose of the file. See extension.

format

Columns contain information in one of four types; users can specify how they want a query to format information it retrieves from character, number, date, or long columns. For example, they can choose to have information of type date appear as 23/11/98, or Monday Twenty-third November 1998, or any other valid date format.

format model

A clause element that controls the appearance of a value in a report column. You specify predefined format models in the COLUMN, TTITLE, and BTITLE commands' FORMAT clauses. You can also use format models for DATE columns in SQL date conversion functions, such as TO_DATE.

form feed

A control character that, when executed, causes the printer to skip to the top of a new sheet of paper (top of form). When SQL*Plus displays a form feed on most terminals, the form feed clears the screen.

formula column

Manually-created column that gets its data from a PL/SQL procedure, function, or expression, user exit, SQL statement, or any combination of these.

function

A PL/SQL subprogram that executes an operation and returns a value at the completion of the operation. A function can be either built-in or user-named. Contrast with procedure.

heading

In SQL*Plus, text that names an output column, appearing above the column. See also column heading.

host computer

The computer from which you run SQL*Plus.

HTML

HTML (HyperText Markup Language) is the language used to write most of the documents available on the World Wide Web. It provides display and linking specifications that are recognized by most web browsers. The HTML recommendation is sponsored by the World Wide Web Consortium (w3) and further details about the w3 and the HTML recommendation can be found at the w3 web site: *http://www.w3.org*.

instance

The background processes and memory area required to access an Oracle database. A database system requires one instance and one database. An Oracle database server consists of an SGA and a set of Oracle database server system processes.

instance failure

See database instance failure.

instance recovery

Recovery of an instance in the event of software or hardware failure, so that the database is again available to users. If the instance terminates abnormally, then the instance recovery automatically occurs at the next instance startup.

Julian date

An algorithm for expressing a date in integer form, using the SQL function JDATE. Julian dates allow additional arithmetic functions to be performed on dates.

justification

See alignment.

label

Defines the label to be printed for the computed value in the COMPUTE command. The maximum length of a COMPUTE label is 500 characters.

LGWR

See Log Writer (LGWR).

local database

The database that SQL*Plus connects to when you start SQL*Plus, ordinarily a database on your host computer. Also called a default database. See also remote database.

log in (or log on)

To perform a sequence of actions at a terminal that establishes a user's communication with the operating system and sets up default characteristics for the user's terminal session.

log off (or log out)

To terminate interactive communication with the operating system, and end a terminal session.

Log Writer (LGWR)

A background process used by an Oracle instance. LGWR writes redo log entries to disk. Redo log data is generated in the redo log buffer of the system global area. As transactions commit and the log buffer fills, LGWR writes redo log entries into an online redo log file.

logon string

A user-specified command line, used to run an application that is connected to either a local or remote database. The logon string either explicitly includes a connect identifier or implicitly uses a default connect identifier.

LONG datatype

One of the standard Oracle datatypes. A LONG column can contain any printable characters such as A, 3, &, or a blank, and can have any length from 0 to 2 gigabytes.

MARKUP

Refers to the SET MARKUP clause or the SQLPLUS -MARKUP clause that permits SQL*Plus output to be generated in HTML format for delivery on the Internet. SQL*Plus output generated in HTML can be viewed with any web browser supporting HTML 3.2.

mounted database

A database associated with an Oracle instance. The database may be opened or closed. A database must be both mounted an opened to be accessed by users. A database that has been mounted but not opened can be accessed by DBAs for some maintenance purposes.

multi-threaded server

Allows many user processes to share a small number of server processes, minimizing the number of server processes and maximizing the utilization of available system resources.

NCHAR datatype

Beginning with Oracle9*i*, the NCHAR datatype is redefined to be a Unicode-only datatype. The NCHAR datatype specifies a fixed-width national character set character string, with width specifications referring to the number of characters, and can have a maximum column size up to 2000 bytes. For more information about the NCHAR datatype, refer to the *Oracle9i SQL Reference*.

NCLOB datatype

A standard Oracle datatype. The NCLOB datatype is used to store fixed-width national character set character (NCHAR) data, and can store up to 4 gigabytes of character text data.

Net8

See Oracle Net.

network

A group of two or more computers linked together through hardware and software to allow the sharing of data and/or peripherals.

NLS_LENGTH_SEMANTICS

NLS_LENGTH_SEMANTICS is an environmental parameter used by the SQL*PLus client application to enable you to create CHAR and VARCHAR2 columns and variables using either byte or character length semantics. NCHAR, NVARCHAR2, CLOB and NCLOB columns are always character-based, and hence are not affected by this variable. If this variable has not been explicitly set at session startup, a default value of BYTE is used (Byte length semantics). The value of NLS_LENGTH_SEMANTICS is then applied as the length semantics of any CHAR or VARCHAR2 declarations which DO NOT explicitly state the length qualifier. NLS_LENGTH_SEMANTICS is also used when displaying variables, or describing tables, views, synonyms, or other objects. On the server side, NLS_LENGTH_SEMANTICS can be set as an initialization parameter, and can be dynamically altered via the 'ALTER SESSION' and 'ALTER SYSTEM' SQL commands. For more information about setting NLS_LENGTH _SEMANTICS on the server side, refer to the *Oracle9i Globalization Support Guide*. Note that NLS_LENGTH_SEMANTICS may differ between the client and server, but the issuing of an ALTER SESSION SET NLS_

LENGTH_SEMANTICS=value command to alter the session scope, will be reflected in the SQL*PLus session.

null

A value that means, “a value is not applicable” or “the value is unknown”. Nulls are not equal to any specific value, even to each other. Comparisons with nulls are always false.

NULL value

The absence of a value.

NUMBER datatype

A standard Oracle datatype. A NUMBER column can contain a number, with or without a decimal point and a sign, and can have from 1 to 105 decimal digits (only 38 digits are significant).

NVARCHAR2 datatype

Beginning with Oracle9*i*, the NVARCHAR2 datatype is redefined to be a Unicode-only datatype. The NVARCHAR2 datatype specifies a variable-width national character set character string, with width specifications referring to the number of characters, and can have a maximum column size up to 4000 bytes. For more information about the NVARCHAR2 datatype, refer to the *Oracle9i SQL Reference*.

object

An object is an instance of an object type. In Oracle9*i*, objects can be persistent (i.e. stored in the database) or transient (i.e. PL / SQL or Oracle Call Interface™ (OCI) variables). See also object type.

object-relational model

A database model that combines the key aspects of the relational and object data models into a single system. Oracle9*i* is an object-relational database system.

object type

A user-defined type that models a structure and behavior of an object. Equivalent to the concept of a class in different programming languages. In Oracle9*i*, object types have public interfaces with attributes and methods. Object types are also known as abstract data types.

online redo log

(1) Redo log files that have not been archived, but are either available to the instance for recording database activity or are filled and waiting to be archived or reused. (2) A set of two or more online redo log files that record all committed changes made to the database.

open database

A database that has been mounted and opened by an instance and is available for access by users. If a database is open, users can access the information it contains. See also mounted database.

operating system

The system software that manages a computer's resources, performing basic tasks such as allocating memory and allowing computer components to communicate.

Oracle Net

Oracle's remote data access software that enables both client-server and server-server communications across any network. Oracle Net supports distributed processing and distributed database capability. Oracle Net runs over and interconnects many communications protocols. Oracle Net is backward compatible with Net8 and SQL*Net version 2.

Oracle Server

The relational database management system (RDBMS) sold by Oracle Corporation. Components of Oracle Server include the kernel and various utilities for use by DBAs and database users.

output

Results of a report after it is run. Output can be displayed on a screen, stored in a file, or printed on paper.

output file

File to which the computer transfers data.

packages

A method of encapsulating and storing related procedures, functions, and other package constructs together as a unit in the database. While packages provide the database administrator or application developer organizational benefits, they also offer increased functionality and database performance.

page

A screen of displayed data or a sheet of printed data in a report.

parallel server

Some hardware architectures (for example, loosely coupled processors) allow multiple computers to share access to data, software, or peripheral devices. With systems that have the parallel server option, Oracle can take advantage of such hardware platforms by running multiple database instances that share a single physical database. In appropriate applications, the Oracle Parallel Server allows access to a single database by the users on multiple machines with increased database performance.

parameter

A substitution variable consisting of an ampersand followed by a numeral (&1, &2, and so on.). You use parameters in a command file and pass values into them through the arguments of the START command.

parameter file

A file used by Oracle 9*i* Server to provide specific values and configuration settings to be used on database startup. For more information about the function of the parameter file, see the *Oracle9i Administrator's Guide*.

password

A secondary identification word (or string of alphanumeric characters) associated with a username. A password is used for data security and known only to its owner. Passwords are entered in conjunction with an operating system login ID, Oracle username, or account name in order to connect to an operating system or software application (such as the Oracle database). Whereas the username or ID is public, the secret password ensures that only the owner of the username can use that name, or access that data.

PL/SQL

The 3GL Oracle procedural language extension of SQL. PL/SQL combines the ease and flexibility of SQL with the procedural functionality of a structured programming language, such as IF...THEN, WHILE, and LOOP. Even when PL/SQL is not stored in the database, applications can send blocks of PL/SQL to the database rather than individual SQL statements, thereby reducing network traffic.

PL/SQL is interpreted and parsed at runtime, it does not need to be compiled.

procedure

A set of SQL and PL/SQL statements grouped together as an executable unit to perform a very specific task. Procedures and functions are nearly identical; the only difference between the two is that functions always return a single value to the caller, while procedures do not return a value to the caller.

process

(1) A thread of control in an operating system; that is, a mechanism in an operating system that can execute a series of steps. Some operating systems use the terms job or task. A process normally has its own private memory area in which it runs.

prompt

(1) A message from a computer program that instructs you to enter data or take some other action. (2) Word(s) used by the system as a cue to assist a user's response. Such messages generally ask the user to respond by typing some information in the adjacent field. See also command line.

query

A SQL SELECT statement that retrieves data, in any combination, expression, or order. Queries are read-only operations; they do not change any data, they only retrieve data. Queries are often considered to be DML statements.

query results

The data retrieved by a query.

RAW datatype

A standard Oracle datatype, a RAW data column may contain data in any form, including binary. You can use RAW columns for storing binary (non-character) data.

RDBMS (Relational Database Management System)

An Oracle7 (and earlier) term. Refers to the software used to create and maintain the system, as well as the actual data stored in the database. See also Relational Database Management System (RDBMS), Server and Oracle Server.

record

A synonym for row; one row of data in a database table, having values for one or more columns.

recover

The Oracle process of restoring all or part of a database from specified redo log files.

redo log

A sequential log of all changes made to the data. The redo log is written and used in the event of a failure that prevents changes from being written to disk. The redo log consists of two or more redo log files.

redo log file

A file containing records of all changes made to the database. These files are used for recovery purposes. See also redo log.

Relational Database Management System (RDBMS)

An Oracle7 (and earlier) term. A computer program designed to store and retrieve shared data. In a relational system, data is stored in tables consisting of one or more rows, each containing the same set of columns. Oracle is a relational database management system. Other types of database systems are called hierarchical or network database systems.

remark

In SQL*Plus, a comment you can insert into a command file with the REMARK command.

remote computer

A computer on a network other than the local computer.

remote database

A database other than your default database, which may reside on a remote computer; in particular, one that you reference in the CONNECT, COPY, and SQLPLUS commands.

report

(1) The results of a query. (2) Any output, but especially output that has been formatted for quick reading, in particular, output from SQL*Plus.

reserved word

(1) A word that has a special meaning in a particular software or operating system. (2) In SQL, a set of words reserved for use in SQL statements; you cannot use a reserved word as the name of a database object.

roles

Named groups of related privileges that are granted to users or other roles.

rollback

To discard pending changes made to the data in the current transaction using the SQL ROLLBACK command. You can roll back a portion of a transaction by identifying a savepoint.

row

(1) Synonym for record; one row of data in a database table, having values for one or more columns. Also called tuple. (2) One set of field values in the output of a query. See also column.

schema

A collection of logical structures of data, or schema objects. A schema is owned by a database user and has the same name as that user.

security level

The combination of a hierarchical classification and a set of non-hierarchical compartments that represent the sensitivity of information.

select

To fetch rows from one or more database tables using a query (the SQL statement SELECT).

SELECT list

The list of items that follow the keyword SELECT in a query. These items may include column names, SQL functions, constants, pseudo-columns, calculations on columns, and aliases. The number of columns in the result of the query will match the number of items in the SELECT list.

SELECT statement

A SQL statement that specifies which rows and columns to fetch from one or more tables or views. See also SQL statement.

Server

Oracle software that handles the functions required for concurrent, shared data access to an Oracle database. The server portion receives and processes SQL and PL/SQL statements originating from client applications. The computer that manages the server portion must be optimized for its duties.

session

The time after a username connects to an Oracle database and before disconnecting, and the events that happen in that time.

SET command variable

See system variable.

SGA

See also System Global Area (SGA).

spooling

Sending or saving output to a disk storage area. Often used in order to print or transfer files. The SQL*Plus SPOOL command controls spooling.

SQL (Structured Query Language)

The internationally accepted standard for relational systems, covering not only query but also data definition, manipulation, security and some aspects of referential integrity. See also Data Manipulation Language (DML), Data Definition Language (DDL), and Data Control Language (DCL).

SQL buffer

The default buffer containing your most recently entered SQL command or PL/SQL block. SQL*Plus commands are not stored in the SQL buffer.

SQL command

See SQL statement.

SQL script

A file containing SQL statements that you can run in SQL*Plus to perform database administration quickly and easily.

SQL statement

A complete command or statement written in the SQL language. Synonymous with statement (SQL).

SQL*Loader

An Oracle tool used to load data from operating system files into Oracle database tables.

SQL*Net

Net8's precursor. An Oracle product that works with the Oracle Server and enables two or more computers that run the Oracle RDBMS or Oracle tools such as SQL*Forms to exchange data through a network. SQL*Net supports distributed processing and distributed database capability. SQL*Net runs over and interconnects many communications protocols.

SQL*Plus

An interactive SQL-based language for data manipulation, data definition and the definition of access rights for an Oracle database. Often used as an end-user reporting tool.

statement (SQL)

A SQL statement, and analogous to a complete sentence, as opposed to a phrase. Portions of SQL statements or commands are called expressions, predicates, or clauses. See also SQL statement.

string

Any sequence of words or characters on a line.

substitution variable

In SQL*Plus, a variable name or numeral preceded by one or two ampersands (&). Substitution variables are used in a command file to represent values to be provided when the command file is run.

subtotal

In a report, a total of values in a number column, taken over a group of rows that have the same value in a break field. See also summary.

summary

Summaries, or summary columns, are used to compute subtotals, grand totals, running totals, and other summarizations of the data in a report.

summary line

A line in a report containing totals, averages, maximums, or other computed values. You create summary lines through the BREAK and COMPUTE commands.

syntax

The orderly system by which commands, qualifiers, and parameters are combined to form valid command strings.

SYS username

See also SYSTEM username.

SYSDBA

Privilege that contains all system privileges with the ADMIN OPTION and the SYSOPER system privilege. See also SYSOPER.

SYSOPER

Privilege that allows a DBA to perform operations such as STARTUP, SHUTDOWN, ARCHIVE LOG and RECOVER. See also SYSDBA.

system administrator

A person responsible for operation and maintenance of the operating system of a computer.

system editor

The text editor provided by the operating system.

System Global Area (SGA)

A shared storage area that contains information required by user processes and background processes, such as data and control information for one Oracle instance.

The SGA is allocated when an Oracle instance is started, and is deallocated when the instance shuts down.

SYSTEM username

One of two standard DBA usernames automatically created with each database (the other is SYS). The Oracle user SYSTEM is created with the password MANAGER. The SYSTEM username is the preferred username for DBAs to use when performing database maintenance.

system variable

A variable that indicates status or environment, which is given a default value by Oracle or SQL*Plus. Examples are LINESIZE and PAGESIZE. Use the SQL*Plus commands SHOW and SET to see and alter the value of system variables.

table

The basic unit of storage in a relational database management system. A table represents entities and relationships, and consists of one or more units of information (rows), each of which contains the same kinds of values (columns).

Each column is given a column name, a datatype (such as CHAR, NCHAR, VARCHAR2, NVARCHAR2, DATE, or NUMBER), and a width (the width may be predetermined by the datatype, as in DATE). Once a table is created, valid rows of data can be inserted into it. Table information can then be queried, deleted, or updated. To enforce defined business rules on a table's data, integrity constraints and triggers can also be defined for a table.

table alias

A temporary substitute name for a table, defined in a query and only good during that query. If used, an alias is set in the FROM clause of a SELECT statement and may appear in the SELECT list. See also alias.

text editor

A program run under your host computer's operating system that you use to create and edit host system files and SQL*Plus command files containing SQL commands, SQL*Plus commands, and / or PL / SQL blocks.

timer

An internal storage area created by the TIMING command.

title

One or more lines that appears at the top or bottom of each report page. You establish and format titles through the TTITLE and BTITLE commands.

transaction

A logical unit of work that comprises one or more SQL statements executed by a single user. According to the ANSI / ISO SQL standard, with which Oracle is compatible, a transaction begins with the user's first executable SQL statement. A transaction ends when it is explicitly committed or rolled back by the user.

truncate

To discard or lose one or more characters from the beginning or end of a value, whether intentionally or unintentionally.

type

A column contains information in one of four types: character, date, number or long. The operations users can perform on the contents of a column depend on the type of information it contains. See also format.

USERID

A command line argument that allows you to specify your Oracle username and password with an optional Oracle Net address.

username

The name by which a user is known to the Oracle database server and to other users. Every username is associated with a private password, and both must be entered to connect to an Oracle database. See also account.

user variable

A variable defined and set by you explicitly with the DEFINE command or implicitly with an argument to the START command.

VARCHAR

An Oracle Corporation datatype. Specifically, this datatype functions identically to the Oracle VARCHAR2 datatype (see definition below). However, Oracle Corporation recommends that you use VARCHAR2 instead of VARCHAR because Oracle Corporation may change the functionality of VARCHAR in the future.

VARCHAR2

An Oracle Corporation datatype. Specifically, it is a variable-length, alpha-numeric string with a maximum length of 4000 bytes. If data entered for a column of type VARCHAR2 is less than 4000 bytes, no spaces will be padded; the data is stored with a length as entered. If data entered is more than 4000 bytes, an error occurs. For more information about the VARCHAR2 datatype, refer to the *Oracle9i SQL Reference*.

variable

A named object that holds a single value. SQL*Plus uses bind substitution, system, and user variables.

view

A view can be thought of as a "stored query" presenting data from one or many tables. A view does not actually contain or store data, but derives data from the base tables on which it is based. Views can be queried, updated, inserted into, and deleted from. Operations on a view affect the view's base tables.

width

The width of a column, parameter, or layout object. Width is measured in characters; a space is a character.

wrapping

A reporting or output feature in which a portion of text is moved to a new line when the entire text does not fit on one line.

Index

-- (comment delimiter), 3-13
- (hyphen)
  clause, 7-2
  continuing a long SQL*Plus command, 2-12, 8-1
. (period), 2-10
/ (slash) command
  default logon, 7-9, 8-46
  entered at buffer line-number prompt, 2-8, 8-9
  entered at command prompt, 2-10, 8-9
  executing current PL/SQL block, 2-10
  executing current SQL command, 2-10
  similar to RUN, 2-10, 8-9, 8-93
  usage, 8-9
# pound sign, 8-34
$ number format, 4-5
& (ampersand)
  substitution variables, 3-23
&&, 3-26
* (asterisk)
  in DEL command, 3-2, 8-54
  in LIST command, 3-2, 8-74
/*...*/ (comment delimiters), 3-12
: (colon)
  bind variables, 3-33
:BindVariable clause
  EXIT command, 8-66
; (semicolon), 2-6
@ ("at" sign)
  command, 3-17, 3-21, 8-5
  command arguments, 8-6
  command file, 3-17, 8-5
  in CONNECT command, 6-3, 8-46
  in COPY command, 6-5, 8-48
  in SQLPLUS command, 3-18, 6-4, 7-2
  passing parameters to a command file, 8-5
  similar to START, 3-17, 8-6, 8-131
@@ (double "at" sign) command, 3-21, 8-7
  command file, 8-7
  similar to START, 8-7, 8-131
[Cancel] key, 2-15
_EDITOR, in EDIT command, 3-8, 8-63
~ infinity sign, 8-34
-~ negative infinity sign, 8-34
0, number format, 4-5
3 tier, B-2
9, number format, 4-5

A

ABORT mode, 8-127
abort query, 2-15
ACCEPT command, 3-30, 8-10
  and DEFINE command, 8-52
  CHAR clause, 8-10
  customizing prompts for value, 3-32
  DATE clause, 8-10
  DEFAULT clause, 8-11
  FORMAT clause, 8-10
  HIDE clause, 8-11
  NOPROMPT clause, 8-11
  NUMBER clause, 3-33
  PROMPT clause, 3-31, 8-11
access, denying and granting, E-2
ALIAS clause, 8-30
  in ATTRIBUTE command, 8-16
ALL clause, 8-122
ALTER command
  disabling, E-5

ampersands (&)
in parameters, 3-29, 8-5, 8-130
substitution variables, 3-23
ANALYZE command
disabling, E-5
APPEND clause
in COPY command, 6-7, 8-49
in SAVE command, 3-19, 8-94
APPEND command, 3-2, 3-6, 8-12
APPINFO clause, 8-98
ARCH background process, 8-14
ARCHIVE LOG
command, 5-4, 8-13
mode, 5-4
argument
in START command, 3-29, 8-130
ARRAYSIZE variable, 8-98
relationship to COPY command, 6-8, 8-50
attribute
display characteristics, 8-16
ATTRIBUTE command, 8-16
ALIAS clause, 8-16
and CLEAR COLUMN command, 8-17
CLEAR clause, 8-16
clearing columns, 8-27, 8-30
controlling display characteristics, 8-17
entering multiple, 8-17
FORMAT clause, 8-17
LIKE clause, 8-17
listing all attributes' display characteristics, 8-16
listing an attribute's display characteristics, 8-16
OFF clause, 8-17
ON clause, 8-17
restoring column display attributes, 8-17
suppressing column display attributes, 8-17
AUDIT command
disabling, E-5
AUTOCOMMIT variable, 2-14, 8-99
AUTOMATIC clause, 8-80
AUTOPRINT variable, 8-99
AUTORECOVERY variable, 8-99
AUTOTRACE variable, 3-39, 8-100
AVG function, 4-16

B

background process
startup after abnormal termination, 8-127
batch mode, 3-20, 8-67
BEGIN command, 2-10
disabling, E-5
bind variables, 3-33
creating, 8-143
displaying, 8-78
displaying automatically, 8-99, 8-145
in PL/SQL blocks, 8-145
in SQL statements, 8-145
in the COPY command, 8-145
REFCURSOR, 3-35
blank line
in PL/SQL blocks, 2-10
in SQL commands, 2-8
preserving in SQL commands, 8-108
blocks, PL/SQL, 1-2
continuing, 2-10
editing in buffer, 3-2
editing with host system editor, 3-8, 8-63
entering and executing, 2-10
listing current in buffer, 3-3
run from SQL buffer, 2-10
saving current, 3-9, 8-94
setting character used to end, 8-100
stored in SQL buffer, 2-10
storing in command files, 3-8
timing statistics, 8-110
within SQL commands, 2-9
BLOCKTERMINATOR, 8-100, 8-108, 8-110
BODY clause, 7-4
BODY option, 7-4
BOLD clause, 8-90, 8-139
break columns, 4-11, 8-18
inserting space when value changes, 4-13
specifying multiple, 4-14
suppressing duplicate values in, 4-12
BREAK command, 4-11, 8-18
and SQL ORDER BY clause, 4-11, 4-12, 4-14, 8-19
clearing BREAKS, 4-15
displaying column values in titles, 4-29

BREAK command (continued)
DUPLICATES clause, 8-21
inserting space after every row, 4-14
inserting space when break column changes, 4-13
listing current break definition, 4-15, 8-21
ON column clause, 4-12, 8-18
ON expr clause, 8-19
ON REPORT clause, 4-19, 8-20
ON ROW clause, 4-14, 8-20
printing "grand" and "sub" summaries, 4-20
printing summary lines at ends of reports, 4-19
removing definition, 8-27
SKIP clause, 4-14, 8-20
SKIP PAGE clause, 4-13, 4-14, 8-21
specifying multiple break columns, 4-14, 8-18
storing current date in variable for titles, 4-31
suppressing duplicate values, 4-12
used in conjunction with COMPUTE, 4-16, 8-18, 8-20, 8-42
used in conjunction with SET COLSEP, 8-101
used to format a REFCURSOR variable, 8-145
break definition
listing current, 4-15, 8-21
removing current, 4-15, 8-27
BREAKS clause, 4-15, 8-27
browser, web, 4-37
BTITLE clause, 8-123
BTITLE command, 4-22, 8-23
aligning title elements, 8-139
BOLD clause, 8-139
CENTER clause, 8-139
COL clause, 8-139
FORMAT clause, 8-139
indenting titles, 8-139
LEFT clause, 8-139
OFF clause, 8-139
old form, F-3
printing blank lines before bottom title, 4-26
referencing column value variable, 8-35
RIGHT clause, 8-139
SKIP clause, 8-139
suppressing current definition, 8-139
TAB clause, 8-139
TTITLE command, 8-23
buffer, 2-5
appending text to a line in, 3-6, 8-12
delete a single line, 3-2
delete the current line, 3-2
delete the last line, 3-2
deleting a range of lines, 3-2, 8-54
deleting a single line, 8-54
deleting all lines, 3-2, 8-27, 8-54
deleting lines from, 3-7, 8-54
deleting the current line, 8-54
deleting the last line, 8-54
executing contents, 2-10, 8-9, 8-93
inserting new line in, 3-5, 8-72
listing a range of lines, 3-3, 8-74
listing a single line, 3-2, 8-74
listing all lines, 3-2, 8-74
listing contents, 3-3, 8-74
listing the current line, 3-2, 8-74
listing the last line, 3-3, 8-74
loading into host system editor, 3-8, 8-63
saving contents, 3-9, 8-94
BUFFER clause, 3-2, 3-10, 8-27
BUFFER variable, F-4

C

CANCEL clause, 8-81, 8-84
cancel query, 2-15
CENTER clause, 4-25, 8-90, 8-139
CHANGE command, 3-2, 3-4, 8-24
CHAR clause, 8-10
VARIABLE command, 8-143
CHAR columns
changing format, 4-6, 8-31
default format, 4-6, 8-31
CLEAR clause, 4-9, 8-30
in ATTRIBUTE command, 8-16
CLEAR command, 8-27
BREAKS clause, 4-15, 8-27
BUFFER clause, 3-2, 3-10, 8-27
COLUMNS clause, 4-9, 8-27
COMPUTES clause, 8-27
SCREEN clause, 3-33, 8-27
SQL clause, 8-28
TIMING clause, 8-28

CLOB clause
VARIABLE command, 8-144
CLOB columns
changing format, 4-6, 8-31
default format, 8-31
setting maximum width, 8-105
setting retrieval position, 8-105
setting retrieval size, 8-105
CLOSECURSOR variable, F-2, F-5
CMDSEP variable, 8-101
COL clause, 4-26, 8-90, 8-139
colons (:)
bind variables, 3-33
COLSEP variable, 8-101
COLUMN command, 4-2, 8-29
ALIAS clause, 8-30
and BREAK command, 8-20
and DEFINE command, 8-52
CLEAR clause, 4-9, 8-30
DEFAULT clause, F-3
displaying column values in bottom titles, 4-30, 8-35
displaying column values in top titles, 4-29, 8-34
entering multiple, 8-36
ENTMAP clause, 8-30
FOLD_AFTER clause, 8-30, 8-31
FOLD_BEFORE clause, 8-31
FORMAT clause, 4-5, 4-6, 8-31
formatting columns, 4-6
formatting NUMBER columns, 4-4, 8-32
HEADING clause, 4-2, 8-34
HEADSEP character, 8-34
JUSTIFY clause, 8-34
LIKE clause, 4-8, 8-34
listing column display attributes, 4-9, 8-29
NEW_VALUE clause, 4-29, 4-31, 8-34
NEWLINE clause, 8-34
NOPRINT clause, 4-29, 8-35
NULL clause, 8-35
OFF clause, 4-9, 8-36
OLD_VALUE clause, 4-30, 8-35
ON clause, 4-9, 8-36
PRINT clause, 8-35
resetting to default display, 4-9, 8-30, F-2
restoring column display attributes, 4-9, 8-36
storing current date in variable for titles, 4-31, 8-37
suppressing column display attributes, 4-9, 8-36
TRUNCATED clause, 4-8, 8-36
used to format a REFCURSOR variable, 8-145
WORD_WRAPPED clause, 4-8, 4-10, 8-36
WRAPPED clause, 4-8, 8-36
column headings
aligning, 8-34
changing, 4-2, 8-34
changing character used to underline, 4-3, 8-111
changing to two or more words, 4-3, 8-34
displaying on more than one line, 4-3, 8-34
suppressing printing in a report, 8-104
when truncated, 8-31
when truncated for CHAR and LONG columns, 4-6
when truncated for DATE columns, 4-6
when truncated for NUMBER columns, 4-4
column separator, 8-101, F-2
columns
assigning aliases, 8-30
computing summary lines, 4-16, 8-40
copying display attributes, 4-8, 8-17, 8-34
copying values between tables, 6-4, 6-9, 8-48
displaying values in bottom titles, 4-30, 8-35
displaying values in top titles, 4-29, 8-34
formatting CHAR, VARCHAR, LONG, and DATE, 8-31
formatting in reports, 4-2, 8-29
formatting MLSLABEL, RAW MLSLABEL, ROWLABEL, 8-31
formatting NUMBER, 4-4, 8-32
listing display attributes for all, 4-9, 8-29
listing display attributes for one, 4-9, 8-29
names in destination table when copying, 6-5, 8-49
printing line after values that overflow, 4-10, 8-107
resetting a column to default display, 4-9, 8-30, F-2
resetting all columns to default display, 4-9, 8-27
restoring display attributes, 4-9, 8-17, 8-36

columns (continued)
setting printing to off or on, 4-29, 8-35
starting new lines, 8-34
storing values in variables, 4-29, 8-34
suppressing display attributes, 4-9, 8-17, 8-36
truncating display for all when value overflows, 4-7, 8-111
truncating display for one when value overflows, 4-8, 8-36
wrapping display for all when value overflows, 4-7, 8-111
wrapping display for one when value overflows, 4-8, 8-36
wrapping whole words for one, 4-10
COLUMNS clause, 4-9, 8-27
comma, number format, 4-5
command file extension, 8-94, 8-110, 8-135
command files, 3-8
aborting and exiting with a return code, 3-20, 8-151, 8-153
allowing end-user input, 3-22
creating with a system editor, 3-11
creating with INPUT and SAVE, 3-10
creating with SAVE, 3-9, 8-94
editing with GET and SAVE, 3-19
editing with host system editor, 3-19, 8-63
in @ ("at" sign) command, 3-17, 8-5
in @@ (double "at" sign) command, 8-7
in EDIT command, 3-19, 8-63
in GET command, 3-16, 8-68
in SAVE command, 3-9, 3-11, 8-94
in SQLPLUS command, 3-18, 7-10
in START command, 3-17, 8-130
including comments in, 3-12, 8-86
including more than one PL/SQL block, 3-11
including more than one SQL command, 3-11
listing names with HOST command, 3-10
nesting, 3-18
passing parameters to, 3-29, 8-5, 8-130
registering, 8-98
retrieving, 3-16, 8-68
running, 3-17, 8-5, 8-130
running a series in sequence, 3-18
running as you start SQL*Plus, 3-18, 7-10
running in batch mode, 3-20, 8-67
running nested, 8-7
saving contents of buffer in, 3-9, 8-94
uniform resource identifier, 8-5, 8-7, 8-130
command prompt
host operating system, 2-2
SET SQLPROMPT, 8-110
SQL*Plus, 2-3
commands, 1-2
collecting timing statistics on, 2-15, 8-136
disabling, E-4
host, running from SQL*Plus, 2-16, 8-70
listing current in buffer, 8-74
re-enabling, E-4
spaces, 2-4
SQL
continuing on additional lines, 2-7
editing in buffer, 3-2
editing with host system editor, 3-8, 8-63
ending, 2-8
entering and executing, 2-6
entering without executing, 2-8
executing current, 2-10, 8-9, 8-93
following syntax, 2-7
list of major, D-1
listing current in buffer, 3-3
saving current, 3-9, 8-94
setting character used to end and run, 8-110
SQL*Plus
abbreviations, 2-11
command summary, 8-2
continuing on additional lines, 2-12, 8-1
editing at command prompt, 3-2
ending, 2-13, 8-1
entering and executing, 2-11
entering during SQL command entry, 8-109
obsolete command alternatives, F-2
stopping while running, 2-15
storing in command files, 3-8
tabs, 2-4
types of, 2-4
variables that affect running, 2-13
writing interactive, 3-22
comments
including in command files, 3-12, 8-86, F-2
using -- to create, 3-13

comments (continued)
using /*...*/ to create, 3-12
using REMARK to create, 3-12, 8-86, F-2
COMMIT clause, 8-66
WHENEVER OSERROR, 8-150
WHENEVER SQLERROR, 8-152
COMMIT command, 2-14
COMPATIBILITY variable, 8-101
COMPUTE command, 4-11, 8-40
AVG function, 4-16
computing a summary on different columns, 4-20
COUNT function, 4-16
LABEL clause, 4-17, 4-19, 8-41
listing all definitions, 4-22, 8-42
MAXIMUM function, 4-16
maximum LABEL length, 8-41
MINIMUM function, 4-16
NUMBER function, 4-16
OF clause, 4-16
ON column clause, 4-16, 8-41
ON expr clause, 8-41
ON REPORT clause, 4-19, 8-41
ON ROW clause, 8-41
printing "grand" and "sub" summaries, 4-20
printing multiple summaries on same column, 4-21
printing summary lines at ends of reports, 4-19
printing summary lines on a break, 4-16
referencing a SELECT expression in OF, 8-41
referencing a SELECT expression in ON, 8-42
removing definitions, 4-22, 8-27
STD function, 4-16
SUM function, 4-16
used to format a REFCURSOR variable, 8-145
VARIANCE function, 4-16
COMPUTES clause, 8-27
CONCAT variable, 3-28, 8-101
CONNECT command, 6-2, 6-3, 8-46
and @ ("at" sign), 6-3, 8-46
changing password, 8-46, 8-47, 8-76
connect identifier, 6-3
service name, 6-3
SYSDBA clause, 7-9, 8-47
SYSOPER clause, 7-9, 8-47
username / password, 6-2, 6-3, 6-4, 8-46
CONNECT command (SQL)
disabling, E-5
connect identifier, 6-3, 8-46
in CONNECT command, 8-46
in COPY command, 8-48
in DESCRIBE command, 8-56
in SQLPLUS command, 7-9
CONTINUE clause
WHENEVER OSERROR, 8-150
WHENEVER SQLERROR, 8-152
continuing a long SQL*Plus command, 2-12, 8-1
COPY command, 6-4, 8-48
and @ ("at" sign), 6-5, 8-48
and ARRAYSIZE variable, 6-8, 8-50
and COPYCOMMIT variable, 6-8, 8-50
and LONG variable, 6-8, 8-50
APPEND clause, 6-7, 8-49
copying data between databases, 6-4
copying data between tables on one database, 6-9
CREATE clause, 6-6, 8-49
creating a table, 6-6, 8-49
destination table, 6-5, 8-49
determining actions, 6-5
determining source rows and columns, 6-6, 8-50
disabling, E-5
FROM clause, 6-5, 8-48
INSERT clause, 6-7, 8-49
inserting data in a table, 6-7, 8-49
interpreting messages, 6-8
mandatory connect identifier, 8-49
naming the source table with SELECT, 6-6, 8-50
query, 6-6, 8-50
referring to another user's table, 6-8
REPLACE clause, 6-6, 8-49
replacing data in a table, 6-6, 8-49
sample command, 6-5, 6-6
service name, 6-5, 6-7, 6-9
specifying columns for destination, 6-5, 8-49
specifying the data to copy, 6-6, 8-50
TO clause, 6-5, 8-48
username / password, 6-5, 6-7, 6-9, 8-48
USING clause, 6-6, 8-50
when a commit is performed, 8-50

COPYCOMMIT variable, 8-102
relationship to COPY command, 6-8, 8-50
COPYTYPECHECK variable, 8-102
COUNT function, 4-16
CREATE clause
in COPY command, 6-6, 8-49
CREATE command
disabling, E-5
entering PL/SQL, 2-9
creating flat files, 4-34
creating the PRODUCT_USER_PROFILE table, E-2
cursor variables, 8-145

D

database
connect identifier, 8-46
mounting, 8-132
opening, 8-133
database administrator, 1-4, 5-2
database changes, saving automatically, 2-14, 8-99
DATABASE clause, 8-82
database files
recovering, 8-80
database name at startup, 8-132
databases
connecting to default, 6-2, 8-46
connecting to remote, 6-3, 8-46
copying data between, 6-4, 8-48
copying data between tables on a single, 6-9
disconnecting without leaving SQL*Plus, 6-2, 8-62
mounting, 5-3
opening, 5-3
recovering, 5-5, 8-80
shutting down, 5-2, 5-3
starting, 5-2
DATAFILE clause, 8-82
DATE clause, 8-10
DATE columns
changing format, 4-6, 8-32, 8-38
default format, 4-6
date, storing current in variable for titles, 4-30, 8-35, 8-37
DB2, 8-102
DBA, 5-2
DBA mode, 8-132
DBA privilege, 8-132
DBMS_APPLICATION_INFO package, 8-98
DECLARE command
disabling, E-5
DECLARE command (PL/SQL), 2-10
DEFAULT clause, 8-11
default logins
created at installation, 1-5
DEFINE command, 3-22, 8-52
and host system editor, 3-8, 8-53
and UNDEFINE command, 3-23, 8-142
CHAR values, 8-52
SET DEFINE ON | OFF, 8-102
substitution variables, 3-26, 8-52
DEFINE variable, 3-28, 8-102
DEL command, 3-2, 3-7, 8-54
using an asterisk, 3-2, 8-54
DELETE command
disabling, E-5
DESCRIBE command (SQL*Plus), 2-17, 8-56
connect_identifier, 8-56
PL/SQL properties listed by, 8-57
table properties listed by, 8-56
DISABLED keyword, disabling commands, E-4
disabling
PL/SQL commands, E-5
SQL commands, E-4
SQL*Plus commands, E-4
DISCONNECT command, 6-2, 8-62
DOCUMENT command
obsolete commands
DOCUMENT, F-2, F-3
REMARK as newer version of, F-4
DOCUMENT variable, F-2, F-5
DROP command
disabling, E-5
DUPLICATES clause, 8-21
dynamic reports, B-2

E

ECHO variable, 3-17, 8-102
EDIT command, 3-8, 8-63
  creating command files with, 3-11
  defining _EDITOR, 3-8, 8-63
  disabling, E-5
  modifying command files, 3-19, 8-63
  setting default file name, 8-103
EDITFILE variable, 8-103
EMBEDDED variable, 8-103
EMP table, 1-4
empty line, displaying, 8-77
Enhancement list, Release 8.1, B-2
entities, HTML, 4-47
ENTMAP, 7-5
ENTMAP clause, 4-47, 7-5, 8-30
error messages, interpreting, 2-18
errors, making line containing current, 3-4
escape characters, definition of, 8-103
ESCAPE variable, 3-28, 8-103
example
  embedded CGI report, 4-42
  interactive HTML report, 4-38, 4-41
EXECUTE command, 8-65
  disabling, E-5
executing
  a CREATE command, 2-9
execution statistics
  including in report, 8-100
EXIT clause
  WHENEVER OSERROR, 8-150
  WHENEVER SQLERROR, 8-152
EXIT command, 2-3, 8-66
  :BindVariable clause, 8-66
  COMMIT clause, 8-66
  disabling, E-5
  FAILURE clause, 8-66
  in a command file, 8-131
  ROLLBACK clause, 8-67
  use with SET MARKUP, 4-38
  WARNING clause, 8-66
exit, conditional, 8-150, 8-152
extension, 8-94, 8-110, 8-135

F

FAILURE clause, 8-66
FEEDBACK variable, 8-103
file extension, 8-94, 8-110, 8-135
file extensions, 3-21
file names
  in @ ("at" sign) command, 8-5
  in @@ (double "at" sign) command, 8-7
  in EDIT command, 8-63
  in GET command, 8-68
  in SAVE command, 3-9, 8-94
  in SPOOL command, 4-34, 8-129
  in SQLPLUS command, 7-10
  in START command, 8-130
files
  command files, 3-8
  flat, 4-34
FLAGGER variable, 8-103
flat file, 4-34
FLUSH variable, 8-104
FOLD_AFTER clause, 8-31
FOLD_BEFORE clause, 8-31
footers
  aligning elements, 8-90
  displaying at bottom of page, 8-87
  displaying system-maintained values, 8-89
  formatting elements, 8-90
  indenting, 8-90
  listing current definition, 8-87
  setting at the end of reports, 4-22
  suppressing definition, 8-90
FORCE clause, 8-132
FORMAT clause, 8-10, 8-31
  in ATTRIBUTE command, 8-17
  in COLUMN command, 4-5, 4-6
  in REPHEADER and REPFOOTER commands, 8-90
  in TTITLE and BTITLE commands, 4-28, 8-139
format models, number, 4-5, 8-33
formfeed, to begin a new page, 4-32, 8-106
FROM clause, 8-81
FROM clause (SQL*Plus), 6-5, 8-48

G

GET command, 3-16, 8-68
  disabling, E-5
  LIST clause, 8-68
  modifying command files, 3-19
  NOLIST clause, 8-68
  retrieving command files, 3-16, 8-68
GLOGIN.SQL, 3-20, 3-41, 3-44, 7-10
  See also LOGIN.SQL
GRANT command, E-2
  disabling, E-5

H

HEAD clause, 7-4
HEAD option, 7-4
headers
  aligning elements, 4-24
  displaying at top of page, 8-89
  displaying system-maintained values, 8-89
  setting at the start of reports, 4-22
  suppressing, 4-24
HEADING clause, 4-2, 8-34
HEADING variable, 8-104
headings
  aligning elements, 8-90
  column headings, 8-104
  formatting elements, 8-90
  indenting, 8-90
  listing current definition, 8-91
  suppressing definition, 8-90
HEADSEP variable, 8-104
  use in COLUMN command, 4-3, 8-34
HELP command, 8-69
help, online, 2-6, 7-12, 8-69
HIDE clause, 8-11
HOST command, 2-16, 8-70
  disabling, E-5
  listing command file names with, 3-10
host operating system
  command prompt, 2-2
  editor, 3-8, 8-63
  file, loading into buffer, 8-68
  running commands from SQL*Plus, 2-16, 8-70
HTML, 4-37
  clause, 7-4
  entities, 4-47
  option, 7-4
  spooling to file, 7-6
  tag, 4-37
hyphen
  continuing a long SQL*Plus command, 2-12, 8-1

I

IMMEDIATE mode, 8-127
infinity sign (~), 8-34
initialization parameters
  displaying, 8-123
INIT.ORA file
  parameter file, 8-132
input
  accepting [Return], 3-33, 8-77
  accepting values from the user, 3-30, 8-10
INPUT command, 3-2, 3-5, 8-72
  entering several lines, 8-72
  using with SAVE to create command files, 3-10
INSERT clause, 6-7, 8-49
INSERT command
  disabling, E-5
installation
  default users created, 1-5
INSTANCE variable, 8-104
instances
  shutting down, 8-127
  starting, 8-132
iSQL*Plus, B-2

J

JUSTIFY clause, 8-34

L

LABEL variable
  SHOW command, F-2, F-7
labels
  in COMPUTE command, 4-17, 8-41
LEFT clause, 4-25, 8-90, 8-139

LIKE clause, 4-8, 8-17, 8-34
limits, SQL*Plus, C-1
line numbers, for SQL commands, 2-6
lines
  adding at beginning of buffer, 8-72
  adding at end of buffer, 8-72
  adding new after current, 3-5, 8-72
  appending text to, 3-6, 8-12
  changing width, 4-32, 8-105
  deleting all in buffer, 8-54
  deleting from buffer, 3-7, 8-54
  determining which is current, 3-4
  editing current, 3-4
  listing all in buffer, 3-2, 8-74
  removing blanks at end, 8-111
LINESIZE variable, 4-24, 4-32, 8-105
LIST clause, 8-13, 8-68
LIST command, 3-2, 8-74
  determining current line, 3-4, 8-74
  making last line current, 3-4, 8-74
  using an asterisk, 3-2, 8-74
LNO clause, 8-123
LOBOFFSET variable, 8-105
LOCK TABLE command
  disabling, E-5
LOG_ARCHIVE_DEST parameter, 8-13
LOGFILE clause, 8-81
logging off
  conditionally, 8-150, 8-152
  Oracle, 6-2, 8-62
  SQL*Plus, 2-3, 8-66
logging on
  default users created at installation, 1-5
  Oracle, 6-2, 6-3, 8-46
  SQL*Plus, 2-3
LOGIN.SQL, 3-20, 7-10
  including SET commands, 3-20
  sample commands to include, 3-20
  See also GLOGIN.SQL
  storing current date in variable for titles, 4-30
LONG columns
  changing format, 4-6, 8-31
  default format, 8-31
  setting maximum width, 8-105
  setting retrieval size, 8-105
LONG variable, 8-105
  effect on COPY command, 6-8, 8-50
LONGCHUNKSIZE variable, 4-6, 8-31, 8-105, 8-106

M

-MARKUP, 4-37, 7-3
  SPOOL clause, 7-4
  SQLPLUS command clause, 7-4
MARKUP, 4-37, 7-3, 8-106
  BODY clause, 7-4
  ENTMAP clause, 7-5
  example, 8-117
  HEAD clause, 7-4
  PREFORMAT clause, 7-6
  SPOOL clause, 7-4
  TABLE clause, 7-4
MAXDATA variable, F-2, F-6
MAXIMUM function, 4-16
media recovery, 8-133
message, sending to screen, 3-30, 8-79
MINIMUM function, 4-16
MOUNT clause, 8-132
mounting a database, 8-132

N

NCHAR clause
  VARIABLE command, 8-143
NCHAR columns
  changing format, 4-6, 8-31
  default format, 4-6, 8-31
NCLOB clause
  VARIABLE command, 8-144
NCLOB columns
  changing format, 4-6, 8-31
  default format, 8-31
  setting maximum width, 8-105
  setting retrieval position, 8-105
  setting retrieval size, 8-105
negative infinity sign (-~), 8-34
NEW_VALUE clause, 4-29, 8-34
  storing current date in variable for titles, 4-31, 8-35

NEWLINE clause, 8-34
NEWPAGE command, F-2, F-4
NEWPAGE variable, 4-31, 8-106
NEXT clause, 8-14
NLS_DATE_FORMAT, 8-10, 8-38
NOAUDIT command
  disabling, E-5
NOLIST clause, 8-68
NOLOG, 7-9
/NOLOG option, 7-9
NOMOUNT clause, 8-133
NONE clause
  WHENEVER OSERROR, 8-150
  WHENEVER SQLERROR, 8-152
NOPRINT clause, 4-17, 4-29, 8-35
NOPROMPT clause, 8-11
NORMAL mode, 8-127
NULL clause, 8-35
null values
  setting text displayed, 8-35, 8-106
NULL variable, 8-106
NUMBER clause, 3-33, 8-10
  VARIABLE command, 8-143
NUMBER columns
  changing format, 4-4, 8-32
  default format, 4-4, 8-33
number formats
  $, 4-5
  0, 4-5
  9, 4-5
  comma, 4-5
  setting default, 8-106
NUMBER function, 4-16
NUMFORMAT clause
  in LOGIN.SQL, 3-20
NUMFORMAT variable, 8-106
NUMWIDTH variable, 8-106
  effect on NUMBER column format, 4-4, 8-33
NVARCHAR2 columns
  changing format, 4-6, 8-31
  default format, 4-6, 8-31

O

objects, describing, 8-102
obsolete commands
  BTITLE, F-3
  COLUMN command DEFAULT clause, F-3
  NEWPAGE, F-2, F-4
  SET command BUFFER variable, F-4
  SET command CLOSECURSOR variable, F-2, F-5
  SET command DOCUMENT variable, F-2, F-5
  SET command MAXDATA variable, F-2, F-6
  SET command SCAN variable, F-2, F-6
  SET command SPACE variable, F-2, F-6
  SET command TRUNCATE variable, F-2, F-7
  SHOW command LABEL variable, F-2, F-7
  TTITLE command old form, F-7
OF clause, 4-16
OFF clause, 8-36
  in ATTRIBUTE command, 8-17
  in COLUMN command, 4-9, 8-36
  in REPFOOTER commands, 8-90
  in REPHEADER commands, 8-90
  in SPOOL command, 4-34, 8-129
  in TTITLE and BTITLE commands, 4-29, 8-139
OLD_VALUE clause, 4-30, 8-35
ON clause
  in ATTRIBUTE command, 8-17
  in COLUMN command, 4-9, 8-36
  in TTITLE and BTITLE commands, 4-29
ON column clause
  in BREAK command, 4-12, 8-18
  in COMPUTE command, 4-16, 8-41
ON expr clause
  in BREAK command, 8-19
  in COMPUTE command, 8-41
ON REPORT clause
  in BREAK command, 4-19, 8-20
  in COMPUTE command, 4-19, 8-41
ON ROW clause
  in BREAK command, 4-14, 8-20
  in COMPUTE command, 8-41
online help, 2-6, 7-12, 8-69
OPEN clause, 8-133
opening a database, 8-133
Oracle Net
  connect identifier, 8-46
  protocol, 6-3

ORDER BY clause
displaying column values in titles, 4-29
displaying values together in output, 4-11
OUT clause, 4-35, 8-129
output
formatting white space in, 8-110
pausing during display, 2-18, 8-107
query results, 1-2

P

PAGE clause, 8-89
page number, including in titles, 4-14, 4-27
pages
changing length, 4-31, 8-106
default dimensions, 4-31
matching to screen or paper size, 4-31
setting dimensions, 4-31
PAGESIZE variable, 2-7, 4-32, 8-106
PAGESIZEclause
in LOGIN.SQL, 3-20
parameter files (INIT.ORA files)
specifying alternate, 8-132
parameters, 3-29, 8-5, 8-130
PARAMETERS clause, 8-123
password, 1-5
changing with the PASSWORD command, 8-76
in CONNECT command, 6-2, 6-3, 8-46
in COPY command, 6-5, 6-7, 6-9
in SQLPLUS command, 2-2, 6-4, 7-9
PASSWORD command, 8-46, 8-76
disabling, E-5
PAUSE command, 3-33, 8-77
in LOGIN.SQL, 3-20
PAUSE variable, 2-18, 8-107
performance
of SQL statements, 3-39
performance, over dial-up lines, 8-111
Period (.)
terminating PL/SQL blocks, 8-100
period (.)
terminating PL/SQL blocks, 2-10
PLAN_TABLE table, 3-39
PL/SQL, 1-2, 2-10
blocks, PL/SQL, 2-10
executing, 8-65
formatting output in SQL*Plus, 8-145
listing definitions, 2-18
mode in SQL*Plus, 2-9
within SQL commands, 2-9
PLUSTRACE role, 3-39
PNO clause, 8-124
pound sign (#), 8-34
PREFORMAT, 7-6
PREFORMAT clause, 7-6
PRINT clause, 8-35
PRINT command, 8-78
printing
bind variables automatically, 8-99
REFCURSOR variables, 8-145
SPOOL command, 8-129
PRODUCT_USER_PROFILE table, E-2
prompt
SET SQLPROMPT, 8-110
PROMPT clause, 3-31, 8-11
PROMPT command, 3-30, 8-79
customizing prompts for value, 3-32
prompts for value
bypassing with parameters, 3-29
customizing, 3-32
through ACCEPT, 3-30
through substitution variables, 3-23
PUPBLD.SQL, E-2

Q

queries, 1-2
in COPY command, 6-6, 8-50
show number of records retrieved, 2-7, 8-103
query execution path
including in report, 8-100
query results, 1-2
displaying on-screen, 2-6
sending to a printer, 4-34, 8-129
storing in a file, 4-34, 8-129
QUIT command, 8-66
See also EXIT

R

record separators, printing, 4-10, 8-107
RECOVER
  clause, 8-133
RECOVER command, 8-80
  AUTOMATIC clause, 8-80
  CANCEL clause, 8-81, 8-84
  CONTINUE clause, 8-81
  DATABASE clause, 8-82
  FROM clause, 8-81
  LOGFILE clause, 8-81
  STANDBY DATABASE clause, 8-82
  STANDBY DATAFILE clause, 8-83
  STANDBY TABLESPACE clause, 8-82, 8-83
  UNTIL CANCEL clause, 8-82
  UNTIL CONTROLFILE clause, 8-83
  UNTIL TIME clause, 8-82
  USING BACKUP CONTROL FILE clause, 8-82
recovery
  RECOVER command, 8-80
RECOVERY command
  and database recovery, 5-5
RECSEP variable, 4-10, 8-107
RECSEPCHAR variable, 4-10, 8-107
redo Log Files
  ARCHIVE LOG command, 8-13
REFCURSOR bind variables, 3-35
  in a stored function, 3-35
REFCURSOR clause
  VARIABLE command, 8-145
RELEASE clause, 8-124
REMARK command, 3-12, 8-86
RENAME command
  disabling, E-5
REPFOOTER clause, 8-124
REPFOOTER command, 4-23, 8-87
  aligning footer elements, 8-90
  BOLD clause, 8-90
  CENTER clause, 8-90
  COL clause, 8-90
  FORMAT clause, 8-90
  indenting report footers, 8-90
  LEFT clause, 8-90
  OFF clause, 8-90
  RIGHT clause, 8-90
  SKIP clause, 8-90
  suppressing current definition, 8-90
  TAB clause, 8-90
REPHEADER clause, 8-124
REPHEADER command, 4-23, 8-89
  aligning header elements, 4-25
  aligning heading elements, 8-90
  BOLD clause, 8-90
  CENTER clause, 8-90
  COL clause, 8-90
  FORMAT clause, 8-90
  indenting headings, 8-90
  LEFT clause, 8-90
  OFF clause, 8-90
  PAGE clause, 8-89
  RIGHT clause, 8-90
  SKIP clause, 8-90
  suppressing current definition, 8-90
  TAB clause, 8-90
REPLACE clause
  in COPY command, 6-6, 8-49
  in SAVE command, 3-19, 8-94
report
  breaks, 8-18
  title, F-2
reports, 1-2
  clarifying with spacing and summary lines, 4-11
  columns, 8-30
  creating bottom titles, 4-22, 8-23, F-2
  creating footers, 8-87
  creating headers, 8-89
  creating headers and footers, 4-22
  creating master / detail, 4-29, 8-35
  creating top titles, 4-22, 8-138, F-2
  displaying, 8-100
  dynamic, B-2
  embedded CGI example, 4-42
  formatting column headings, 4-2, 8-29
  formatting columns, 4-4, 4-6, 8-29
  interactive HTML example, 4-38, 4-41
  on the web, 4-37
  SILENT mode, 4-47
  starting on a new page, 8-103
  title, 8-138

RESTRICT, 7-7, 8-132, E-8
return code, specifying, 3-20, 8-67, 8-153
REVOKE command, E-2
  disabling, E-5
RIGHT clause, 4-25, 8-90, 8-139
roles, E-6
  disabling, E-7
  re-enabling, E-8
ROLLBACK clause, 8-67
  WHENEVER OSERROR, 8-150
  WHENEVER SQLERROR, 8-152
rows
  performing computations on, 4-16, 8-40
  setting number retrieved at one time, 8-98
  setting the number after which COPY commits, 8-102
RUN command, 2-10, 8-93
  disabling, E-5
  executing current PL/SQL block, 2-10
  executing current SQL command or PL/SQL block, 2-10
  making last line current, 3-4
  similar to / (slash) command, 2-10, 8-93

S

sample tables
  access to, 1-6
SAVE command, 3-9, 8-94
  APPEND clause, 3-19, 8-94
  CREATE clause, 8-94
  disabling, E-5
  modifying command files, 3-19
  REPLACE clause, 3-19, 8-94
  storing commands in command files, 3-9, 8-94
  using with INPUT to create command files, 3-11
saving environment attributes, 8-135
SCAN variable, F-2, F-6
SCREEN clause, 3-33, 8-27
screen, clearing, 3-33, 8-27
Security
  RESTRICT, 7-7, E-8
security
  changing password, 8-76
  embedded web reports, 4-42
  PRODUCT_USER_PROFILE table, E-2
SELECT command
  and BREAK command, 4-11, 8-19, 8-20
  and COLUMN command, 8-30
  and COMPUTE command, 4-11
  and COPY command, 6-6, 8-50
  and DEFINE command, 8-52
  and ORDER BY clause, 4-11
  disabling, E-5
  storing current date in variable for titles, 4-31
SELECT statement
  formatting results, 3-35
semicolon (;)
  in PL/SQL blocks, 2-10
  in SQL commands, 2-6, 2-8
  in SQL*Plus commands, 2-13, 8-1
  not needed when inputting a command file, 3-11
  not stored in buffer, 3-3
SERVEROUTPUT variable, 8-107
service Name
  in COPY command, 6-9
service name
  in CONNECT command, 6-3
  in COPY command, 6-5, 6-7
  in SQLPLUS command, 6-4
SET AUTOTRACE, 3-39
SET clause, 8-135
SET command, 2-13, 3-21, 8-96
  APPINFO variable, 8-98
  ARRAYSIZE variable, 6-8, 8-98
  AUTOCOMMIT variable, 2-14, 8-99
  AUTOPRINT variable, 8-99, 8-145
  AUTORECOVERY variable, 8-99
  AUTOTRACE variable, 8-100
  BLOCKTERMINATOR variable, 8-100
  BUFFER variable, F-4
  CLOSECURSOR variable, F-2, F-5
  CMDSEP variable, 8-101
  COLSEP variable, 4-34, 8-101
  COMPATIBILITY variable, 8-101
  CONCAT variable, 3-28, 8-101
  COPYCOMMIT variable, 6-8, 8-102
  COPYTYPECHECK variable, 8-102
  DEFINE clause, 3-28

SET command (continued)
DEFINE variable, 8-102
DESCRIBE variable, 8-102
disabling, E-5
DOCUMENT variable, F-2, F-5
ECHO variable, 3-17, 8-102
EDITFILE variable, 8-103
EMBEDDED variable, 8-103
ESCAPE variable, 3-28, 8-103
FEEDBACK variable, 8-103
FLAGGER variable, 8-103
FLUSH variable, 8-104
HEADING variable, 8-104
HEADSEP variable, 4-3, 8-104
INSTANCE variable, 8-104
LINESIZE variable, 4-24, 4-32, 8-105
LOBOFFSET variable, 8-105
LOGSOURCE variable, 8-105
LONG variable, 6-8, 8-105
LONGCHUNKSIZE variable, 8-105
MARKUP clause, 8-106
MAXDATA variable, F-2, F-6
NEWPAGE variable, 4-31, 8-106
NULL variable, 8-106
NUMFORMAT clause, 3-20
NUMFORMAT variable, 8-106
NUMWIDTH variable, 4-4, 8-33, 8-106
PAGESIZE clause, 3-20
PAGESIZE variable, 2-7, 4-32, 8-106
PAUSE clause, 3-20
PAUSE variable, 2-18, 8-107
RECSEP variable, 4-10, 8-107
RECSEPCHAR variable, 4-10, 8-107
SCAN variable, F-2, F-6
SERVEROUTPUT variable, 8-107
SHIFTINOUT variable, 8-108
SPACE variable, F-2, F-6
SQLBLANKLINES variable, 8-108
SQLCASE variable, 8-108
SQLCONTINUE variable, 8-109
SQLNUMBER variable, 8-109
SQLPLUSCOMPATIBILITY variable, 8-109, B-3
SQLPREFIX variable, 8-109
SQLPROMPT variable, 8-110
SQLTERMINATOR variable, 8-110
SUFFIX variable, 8-110
TAB variable, 8-110
TERMOUT variable, 4-31, 8-110
TIME clause, 3-21
TIME variable, 8-110
TIMING variable, 8-110
TRIMOUT variable, 8-111
TRIMSPOOL variable, 8-111
TRUNCATE variable, F-2, F-7
UNDERLINE variable, 4-3, 8-111
used to format a REFCURSOR variable, 8-145
VERIFY clause, 3-24
VERIFY variable, 3-28, 8-111
WRAP variable, 4-7, 8-111
SET command variables
system variables, 2-13
SET MARKUP
BODY clause, 7-4
embedded CGI example, 4-42
ENTMAP clause, 4-47, 7-5
HEAD clause, 7-4
HTML, 7-4
interactive HTML example, 4-38, 4-41
PREFORMAT clause, 7-6
See also SPOOL command
SPOOL clause, 7-4
TABLE clause, 7-4
SET ROLE command
disabling, E-5
SET TRANSACTION command
disabling, E-5
SGA clause, 8-124
SHIFTINOUT variable, 8-108
SHOW clause, 8-136
SHOW command, 2-13, 8-122
ALL clause, 8-122
BTITLE clause, 8-123
ERRORS clause, 8-123
LABEL variable, F-2, F-7
listing current page dimensions, 4-33
LNO clause, 8-123
PNO clause, 8-124
RELEASE clause, 8-124
REPFOOTER clause, 8-124
REPHEADER clause, 8-124

SHOW command (continued)
SPOOL clause, 8-124
SQLCODE clause, 8-124
TTITLE clause, 8-124
USER clause, 8-124
SHOWMODE variable, 8-108
SHUTDOWN command, 8-127
ABORT, 8-127
IMMEDIATE, 8-127
NORMAL, 8-127
-SILENT option, 4-47, 7-8
site profile
GLOGIN.SQL, 7-10
LOGIN.SQL, 7-10
See also user profile
SKIP clause
in BREAK command, 4-13, 4-14, 8-20
in REPHEADER and REPFOOTER commands, 8-90
in TTITLE and BTITLE commands, 4-25, 8-139
used to place blank lines before bottom title, 4-26
SKIP PAGE clause, 4-13, 4-14, 8-21
slash (/) command, 8-9
using with files loaded with GET command, 8-68
SPACE variable, F-2, F-6
SPOOL clause, 7-4, 7-5, 8-124
SPOOL command, 4-33, 8-129
disabling, E-5
file name, 4-34, 8-129
OFF clause, 4-34, 8-129
OUT clause, 4-35, 8-129
to HTML file, 7-6
turning spooling off, 4-34, 8-129
use with SET MARKUP, 4-38
SQL buffer, 2-5
SQL clause, 8-28
SQL commands, list of major, D-1
SQL database language, 1-2
SQL DML statements
reporting on, 8-100
SQL*Plus
basic concepts, 1-2
command prompt, 2-3
command summary, 8-2
database administration, 5-2
exiting, 2-3, 8-66
exiting conditionally, 8-150, 8-152
limits, C-1
LOGIN.SQL, 3-20
obsolete command alternatives, F-2
overview, 1-2
running commands in batch mode, 3-20, 8-67
setting up environment, 3-20
shortcuts to starting, 2-3
starting, 2-2, 7-2
what you need to run, 1-4
who can use, 1-3
SQLBLANKLINES variable, 8-108
SQLCASE variable, 8-108
SQLCODE clause, 8-124
SHOW command, 8-124
SQLCONTINUE variable, 8-109
SQLNUMBER variable, 8-109, B-3
SQLPLUS command, 2-2, 7-2
- clause, 7-2
-? clause, 7-2
and @ ("at" sign), 3-18, 6-4, 7-2
and EXIT FAILURE, 7-10
BODY option, 7-4
connect identifier, 7-9
connecting to a remote database, 6-4
display syntax, 7-2
ENTMAP option, 7-5
HEAD option, 7-4
HTML option, 7-4
-MARKUP clause, 7-4
-MARKUP option, 7-3
-MARKUP SPOOL clause, 7-4
/ NOLOG clause, 7-9
PREFORMAT option, 7-6
RESTRICT, 7-7, E-8
running command files, 3-18
service name, 6-4, 7-9
-SILENT clause, 7-8
-SILENT option, 4-47, 7-8
SPOOL clause, 7-5
syntax, 7-2
SYSDBA clause, 7-9

SQLPLUS command (continued)
TABLE option, 7-4
unsuccessful connection, 7-10
username/password, 2-2, 7-9
SQL.PNO, referencing in report titles, 4-27
SQLPREFIX variable, 8-109
SQLPROMPT variable, 8-110
SQL.SQLCODE
using in EXIT command, 8-66
SQLTERMINATOR, 8-108
SQLTERMINATOR variable, 8-70, 8-110
STANDBY DATAFILE clause, 8-83
STANDBY TABLESPACE clause, 8-82
START clause, 8-14, 8-136
START command, 3-17, 8-130
arguments, 3-29, 8-130
command file, 3-17, 8-130
disabling, E-5
passing parameters to a command file, 3-29, 8-130
similar to @ ("at" sign) command, 3-17, 8-6, 8-131
similar to @@ (double "at" sign) command, 8-7, 8-131
starting SQL*Plus, 2-2
shortcuts, 2-3
STARTUP command, 8-132
FORCE clause, 8-132
MOUNT clause, 8-132
NOMOUNT clause, 8-133
OPEN clause, 8-133
PFILE clause, 8-132
RECOVER clause, 8-133
RESTRICT clause, 8-132
specifying a database, 8-132
statistics, 3-41
STD function, 4-16
STOP clause, 8-14, 8-136
stop query, 2-15
STORE command, 3-21, 8-135
SET clause, 8-135
stored functions, 3-35
stored procedures
creating, 2-9
substitution variables, 3-23
appending characters immediately after, 3-25
avoiding unnecessary prompts for value, 3-26
concatenation character, 8-101
DEFINE command, 3-26, 8-52
defined, 3-26
prefixing, 8-102, F-2
restrictions, 3-28
single and double ampersands, 3-26
system variables used with, 3-28
undefined, 3-23
where and how to use, 3-23
SUFFIX variable, 8-110
used with @ ("at" sign) command, 8-5
used with EDIT command, 8-63
used with GET command, 8-68
used with SAVE command, 8-94
used with START command, 8-130
SUM function, 4-16
summary lines
computing and printing, 4-16, 8-40
computing and printing at ends of reports, 4-19
computing same type on different columns, 4-20
printing "grand" and "sub" summaries (totals), 4-20
printing multiple on same break column, 4-21
syntax
COPY command, 6-5
syntax rules
SQL commands, 2-7
SQL*Plus commands, 2-12
SYSDATE, 4-31
SYSDBA clause, 8-47
SYSOPER clause, 7-9, 8-47
system variables, 2-13, 8-111
changing current settings, 8-96
listing current settings, 2-13, 8-122
listing old and new values, 8-108
storing and restoring, 3-21
used with substitution variables, 3-28
system-maintained values
displaying in headers and footers, 8-89
displaying in titles, 4-27, 8-138
formatting in titles, 4-28

T

TAB clause, 8-90, 8-139
TAB variable, 8-110
TABLE clause, 7-4
TABLE option, 7-4
tables, 1-2
  access to sample, 1-6
  controlling destination when copying, 6-6, 8-49
  copying values between, 6-4, 6-9, 8-48
  listing column definitions, 2-17, 8-56
  referring to another user's when copying, 6-8
TABLESPACE clause, 8-82
tablespaces
  recovering, 8-80
tag, HTML, 4-37
TERMOUT variable, 8-110
  storing current date in variable for titles, 4-31
  using with SPOOL command, 8-129
text, 7-4
  adding to current line with APPEND, 3-6, 8-12
  changing old to new with CHANGE, 3-4, 8-24
  clearing from buffer, 3-2, 8-27
text editor, host operating system, 3-8, 8-63
three-tier architecture, B-2
TIME clause
  in LOGIN.SQL, 3-21
TIME variable, 8-110
TIMING clause, 8-28
TIMING command, 2-15, 8-136
  deleting all areas created by, 8-28
  deleting current area, 8-136
  SHOW clause, 8-136
  START clause, 8-136
  STOP clause, 8-136
TIMING variable, 8-110
titles
  aligning elements, 4-24, 8-139
  displaying at bottom of page, 4-22, 8-23, F-2
  displaying at top of page, 4-22, 8-138, F-2
  displaying column values, 4-29, 8-34, 8-35
  displaying current date, 4-30, 8-35, 8-37
  displaying page number, 4-27, 8-140
  displaying system-maintained values, 4-27, 8-138
  formatting elements, 8-139
  formatting system-maintained values in, 4-28
  indenting, 4-26, 8-139
  listing current definition, 4-28, 8-23, 8-140
  restoring definition, 4-29
  setting at start or end of report, 4-22
  setting lines from top of page to top title, 4-31, 8-106, F-2
  setting lines from top title to end of page, 8-106
  setting top and bottom, 4-22, 8-23, 8-138, F-2
  spacing between last row and bottom title, 4-25
  suppressing definition, 4-29, 8-139
TO clause, 6-5, 8-48
tracing statements, 3-39
  for performance statistics, 3-41
  for query execution path, 3-41
  using a database link, 3-43
  with parallel query option, 3-44
TRIMOUT variable, 8-111
TRIMSPOOL variable, 8-111
TRUNCATE command
  disabling, E-5
TRUNCATE variable, F-2, F-7
TRUNCATED clause, 4-8, 8-36
TTITLE clause, 8-124
TTITLE command, 4-22, 8-138
  aligning title elements, 4-24, 8-139
  BOLD clause, 8-139
  CENTER clause, 4-25, 8-139
  COL clause, 4-26, 8-139
  FORMAT clause, 4-28, 8-139
  indenting titles, 4-26, 8-139
  LEFT clause, 4-25, 8-139
  listing current definition, 4-28, 8-140
  OFF clause, 4-29, 8-139
  old form, F-7
  ON clause, 4-29
  referencing column value variable, 4-29, 8-34
  restoring current definition, 4-29
  RIGHT clause, 4-25, 8-139
  SKIP clause, 4-25, 8-139
  suppressing current definition, 4-29, 8-139
  TAB clause, 8-139
tuning, 3-39

U

UNDEFINE command, 3-23, 8-142
  and DEFINE command, 8-52
UNDERLINE variable, 4-3, 8-111
UNTIL CANCEL clause, 8-82
UNTIL CHANGE clause, 8-82
UNTIL CONTROLFILE clause, 8-83
UNTIL TIME clause, 8-82
UPDATE command
  disabling, E-5
USER clause, 8-124
user profile, 3-20
  LOGIN.SQL, 7-10
  See also site profile
user variables, 3-22
  defining, 3-22, 8-52
  deleting, 3-23, 8-142
  displaying in headers and footers, 8-89
  displaying in titles, 8-138
  in ACCEPT command, 3-30, 8-10
  listing definition of one, 3-22, 8-52
  listing definitions of all, 3-23, 8-52
username, 1-5
  connecting under different, 6-2, 8-46
  created at installation, 1-5
  in CONNECT command, 6-2, 6-3, 8-46
  in COPY command, 6-5, 6-7, 6-9
  in SQLPLUS command, 2-2, 6-4, 7-9
userprofile
  GLOGIN.SQL, 7-10
USING BACKUP CONTROL FILE clause, 8-82
USING clause, 6-6, 8-50

V

V$SESSION virtual table, 8-98
V$SQLAREA virtual table, 8-98
VARCHAR columns
  changing format, 4-6
  default format, 4-6, 8-31
VARCHAR2 clause
  VARIABLE command, 8-144
VARCHAR2 columns
  changing format, 4-6, 8-31
  default format, 4-6
VARIABLE command, 8-143
  CHAR clause, 8-143
  CLOB clause, 8-144
  NCHAR clause, 8-143
  NCLOB clause, 8-144
  NUMBER clause, 8-143
  REFCURSOR clause, 8-145
  VARCHAR2 clause, 8-144
  variable clause, 8-143
variables
  bind variables, 3-33
  substitution variables, 3-23
  system variables, 2-13
  user variables, 8-52
VARIANCE function, 4-16
VERIFY clause, 3-24
VERIFY variable, 3-28, 8-111

W

WARNING clause, 8-66
web
  outputting reports, 4-37
web browser, 4-37
WHENEVER OSERROR command, 8-150
  COMMIT clause, 8-150
  CONTINUE clause, 8-150
  EXIT clause, 8-150
  NONE clause, 8-150
  ROLLBACK clause, 8-150
WHENEVER SQLERROR command, 3-20, 8-152
  COMMIT clause, 8-152
  CONTINUE clause, 8-152
  EXIT clause, 8-152
  NONE clause, 8-152
  ROLLBACK clause, 8-152
WORD_WRAPPED clause, 4-8, 4-10, 8-36
WRAP variable, 4-7, 8-111
WRAPPED clause, 4-8, 8-36