How-To Learn and Use SQL*Plus Basics

Oracle Database

Entering and Executing Commands

Unless stated otherwise, descriptions of commands are applicable to all user interfaces.

In the command-line, type commands at the SQL*Plus prompt and press Return to execute them.

Usually, you separate the words in a command with a space or a tab. You can use additional spaces or tabs between words to make your commands more readable.

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:

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 SQL buffer stores the most recently entered SQL command or PL/SQL block (but not SQL*Plus commands). The command or block remains in the buffer until replaced by the next SQL command or PL/SQL block. You can view the buffer contents with the LIST command.

You can execute the command or block in the SQL buffer using the RUN or /(slash) commands. RUN displays the command or block in the buffer before executing it. /(slash) executes the command or block in the buffer without displaying it first. For information about editing a command or block stored in the buffer see Editing Scripts in SQL*Plus Command-Line.

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

Executing Commands

In command-line SQL*Plus, you type a command and direct SQL*Plus to execute it by pressing the Return key. SQL*Plus processes the command and re-displays the command prompt when ready for another command.

Listing a Table Definition

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

Example 4-1 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 Database data dictionary. You can also use SQL SELECT commands to access this and other information in the database. See your Oracle Database SQL Language Reference for details.

Listing PL/SQL Definitions

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

Example 4-2 Using the DESCRIBE Command

To create and list the definition of a function called AFUNC, enter

```
create or replace function afunc (f1 varchar2, f2 number) return number as
begin
   if (length(f1) > f2) then
    return 1;
   else
    return 0;
   end if;
end;
/
FUNCTION created.
```

```
DESCRIBE afunc
FUNCTION afunc RETURNS NUMBER
Argument Name   Type     In/Out   Default?
--------------  --------  --------  ---------
F1                  VARCHAR2 IN
F2                  NUMBER    IN
```

Running SQL Commands

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

Example 4-3 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.

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.

SQL*Plus displays 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 or click Execute. SQL*Plus processes the command and displays the results:

```
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 command-line displays the command prompt again. If you made a mistake and therefore did not get the results shown, 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 can use the system variable, LINESIZE, to set the width of the output in characters. See Setting Page Dimensions. Typically, LINESIZE is set to 80 in command-line. Whether you see the message stating the number of records retrieved depends on the setting of the system variable, FEEDBACK. See System Variables that Affect How Commands Run for more information.

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. Thus, you can enter the query you entered in Example 4-3, "Entering a SQL Command" 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 4-3, "Entering a SQL Command", 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 4-3, "Entering a SQL Command", and press Return or click Execute. SQL*Plus processes the command and also stores the command in the SQL buffer. See 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 or click Execute to run the command.

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 or click Execute. SQL*Plus executes the command and stores it in the buffer.

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.

To execute commands this way, 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 SQL Buffer for details. If you subsequently enter another SQL command, SQL*Plus overwrites the previous command in the buffer.

Running PL/SQL Blocks

You can also use PL/SQL subprograms (called blocks) to manipulate data in the database. See your Oracle Database PL/SQL Language Reference for information on individual PL/SQL statements.

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.

You enter the mode for entering PL/SQL statements when:

You type DECLARE or BEGIN. 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 PROCEDURE) that creates a stored procedure. After you enter PL/SQL mode in this way, type the stored procedure you want to create.

SQL*Plus stores the subprograms you enter in the SQL buffer. Execute the current subprogram with a RUN or slash (/) command. A semicolon (;) is treated as part of the PL/SQL subprogram and will not execute the command.

SQL*Plus sends the complete PL/SQL subprogram to Oracle Database for processing (as it does SQL commands). See your Oracle Database PL/SQL Language 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;
  .
/
```

Creating Stored Procedures

Stored procedures are PL/SQL functions, packages, or procedures. To create stored procedures, you use the following SQL CREATE commands:

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 Running PL/SQL Blocks. 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

See SHOW for more information.

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

EXECUTE EMPLOYEE_MANAGEMENT.NEW_EMP('BLAKE')

See EXECUTE for more information.

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.

To speed up command entry, you can abbreviate many SQL*Plus commands. For information on and abbreviations of all SQL*Plus commands, see SQL*Plus Command Reference.

Example 4-4 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.

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.

Enter the following query and press Return to run it:

SELECT EMPLOYEE_ID, LAST_NAME, JOB_ID, SALARY

FROM EMP_DETAILS_VIEW WHERE SALARY > 12000;

```
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.

Understanding SQL*Plus Command Syntax

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

You do not need to end a SQL*Plus command with a semicolon. When you finish entering the command, you can just press Return or click Execute. There is no need to end a SQL*Plus command with a semicolon.

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.

System Variables that Affect How Commands Run

The SQL*Plus SET command controls many variables—called SET variables or system variables—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.

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.)

See the SET command for more information on system variables and their default settings. See SQL*Plus Configuration and SQLPLUS Program Syntax for details on the SQL*Plus LOGIN file.

To list the current setting of a system variable, enter SHOW followed by the variable name. See the SHOW command for information on other items you can list with SHOW.

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.

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 Formatting SQL*Plus Reports.) 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.

Running Operating System Commands

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

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

HOST DIRECTORY *.SQL

When the 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 affect the current SQL*Plus session. For example, setting an operating system environment variable does not affect the current SQL*Plus session, but may affect SQL*Plus sessions started subsequently.

You can suppress access to the HOST command. For more information about suppressing the HOST command see SQL*Plus Security.

Pausing the Display

You can use the PAUSE system variable to stop and examine the contents of the screen after each page during the display of a long report, or during the display of a table definition with many columns.

You can use SET PAUSE to pause output after displaying each screen of a query or report. See SET PAU[SE] {ON | OFF | text} for more information.

Saving Changes to the Database Automatically

You can specify changes you wish to make to the information stored in the database using the SQL Database Manipulation Language (DML) commands UPDATE, INSERT, and DELETE—which can be used independently or within a PL/SQL block. These changes are not made permanent until you enter a SQL COMMIT command or a SQL Database Control Language (DCL) or Database Definition Language (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 AUTOCOMMIT system variable. Regardless of the AUTOCOMMIT setting, changes are committed when you exit SQL*Plus successfully.

See Also:

SET EXITC[OMMIT] {ON | OFF}

Example 4-5 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

See SET AUTO[COMMIT]{ON | OFF | IMM[EDIATE] | n} for more information.

Interpreting Error Messages

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

Example 4-6 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 SQL*Plus Error Messages.

If the error is a numbered error beginning with the letters "CPY" look up the SQL*Plus COPY command message in COPY Command Messages.

If the error is a numbered error beginning with the letters "ORA", look up the Oracle Database message in the Oracle Database Error Messages guide or in the platform-specific Oracle documentation provided for your operating system.

If the error is a numbered error beginning with the letters "PLS", look up the Oracle Database message in the Oracle Database PL/SQL Language Reference.

If the error is unnumbered, look up correct syntax for the command that generated the error in SQL*Plus Command Reference for a SQL*Plus command, in the Oracle Database SQL Language Reference for a SQL command, or in the Oracle Database PL/SQL Language Reference for a PL/SQL block. Otherwise, contact your DBA.

Courtesy: https://docs.oracle.com/cd/E11882_01/server.112/e16604/ch_four.htm#i1039736

Modified: 2021.10.09.11.34.AM

Dököll Solutions, Inc.