

*SQL*Plus Syntax & Solutions*

Includes Oracle8i



Oracle SQL*Plus

The Definitive Guide

O'REILLY®

Jonathan Gennick

Oracle Ex. 1029, p. 1
Oracle v. DeBellis
IPR2024-00221

Oracle SQL*Plus: The Definitive Guide

by Jonathan Gennick

Copyright © 1999 O'Reilly & Associates, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly & Associates, Inc. 101 Morris Street, Sebastopol, CA 95472.

Editor: Deborah Russell

Production Editor: Ellie Fountain Maden

Printing History:

March 1999: First Edition.

ORACLE, Oracle, Oracle Enterprise Manager, Oracle Personal Edition, Oracle Procedure Builder, Oracle Server, Oracle Server Manager, Oracle Version 6, Oracle7, Oracle8, Oracle 8i, Net8, SQL*Loader, SQL*Net, and SQL*Plus are trademarks of Oracle Corporation.

PowerBuilder is a trademark of PowerSoft.

SQL Station is a trademark of Platinum Technology.

Clear Access is a trademark of Sterling Software.

Microsoft and Windows NT are registered trademarks of Microsoft Corporation.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly & Associates, Inc. was aware of the trademark claim, the designations have been printed in caps or initial caps.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly & Associates, Inc. The use of the moving leaf image in association with Oracle SQL*Plus is a trademark of O'Reilly & Associates, Inc.

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 1-56592-578-5

[M]

[2/00]

In this chapter:

- *What Is SQL*Plus?*
- *History of SQL*Plus*
- *Why Master SQL*Plus?*
- *Creating and Loading the Sample Tables*

1

*Introduction to SQL*Plus*

SQL*Plus is the command-line interface to the Oracle database. Its fundamental reason for existence is to allow you to enter and execute ad hoc SQL statements and PL/SQL code blocks. This chapter explains what SQL*Plus is, how it relates to other Oracle tools (as well as the database), and why you should master it. At the end of the chapter I'll introduce you to the sample data, which is used for many of the examples in this book. If you like, you can load that data into your database and test out each example as you go through this book.

*What Is SQL*Plus?*

SQL*Plus is essentially an interactive query tool, with some scripting capabilities. It is a non-GUI, character-based tool that has been around since the dawn of the Oracle age. Using SQL*Plus, you can enter an SQL statement, such as a SELECT query, and view the results. You can also execute Data Definition Language (DDL) commands that allow you to maintain and modify your database. You can even enter and execute PL/SQL code. In spite of SQL*Plus's age and lack of "flash," it is a workhorse tool used day in and day out by database administrators, developers, and yes, even end users. As a database administrator, it is my tool of choice for managing the databases under my care. I use it to peek under the hood—to explore the physical implementation of my database, and to create and manage users, tables, and tablespaces. In my role as a developer, SQL*Plus is the first tool that I fire up when I need to develop a query. In spite of all the fancy, GUI-based SQL generators contained in products such as PowerBuilder, Clear Access, and Crystal Reports, I still find it quicker and easier to build up and test a complex query in SQL*Plus before transferring it to whatever development tool I am using.

There are chapters in this book covering each of the above topics in detail. A lot can be accomplished with SQL*Plus. This book will show you how.

*SQL*Plus's Relation to SQL, PL/SQL and the Oracle Database*

SQL*Plus is often used in conjunction with two other products, both of which have the letters “SQL” in their names. The first is SQL itself. Without a doubt, the most common use of SQL*Plus is to submit SQL statements to the database for execution. The second product is Oracle’s PL/SQL procedural language. Table 1-1 gives a short summary of each of these three products.

*Table 1-1. The Three SQLs: SQL, PL/SQL, and SQL*Plus*

Product	Description
SQL	SQL, which stands for Structured Query Language, is an ANSI (and ISO) standard language used for querying, modifying, and managing relational databases. It is used to insert, delete, update, and retrieve data.
PL/SQL	PL/SQL is a proprietary procedural language developed by Oracle as an extension to SQL. Like SQL, it also executes inside the database. It was created as a tool for coding business rules and procedures at the database level.
SQL*Plus	SQL*Plus is an Oracle-developed tool that allows you to interactively enter and execute SQL commands and PL/SQL blocks.

Because these three products all have “SQL” as part of their names, people occasionally get confused about the relationship between them and about which commands get executed where. SQL*Plus does have its own set of commands it recognizes and executes, but any SQL queries, DDL commands, and PL/SQL blocks are sent to the database server for execution. Figure 1-1 illustrates this relationship.

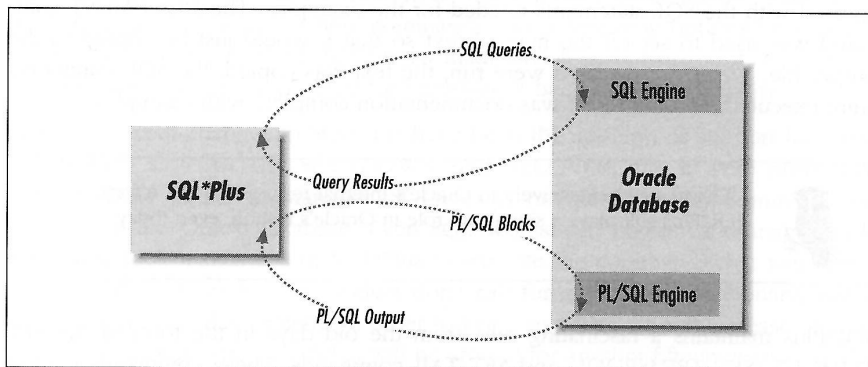


*Figure 1-1. Relationship between SQL*Plus, SQL, and PL/SQL*

B

*SQL*Plus Format Elements*

Several SQL*Plus commands allow you to control data formats using what is called a format specification. A *format specification* is a string of characters that tells SQL*Plus exactly how to format a number, date, or text string when it is displayed. The most notable of these commands is the COLUMN command, which is used to format columns of output from a SELECT query. There are other commands as well. The complete list of SQL*Plus commands that accept format specification strings is shown here:

ACCEPT

Prompts the user to enter a value from the keyboard.

COLUMN

Controls various aspects of the way a column of data is displayed.

SET NUMBER

Defines the default display format for numbers.

TTITLE, BTITLE, REPHEADER, REPFOOTER

These commands all allow number format specifications to control the way numbers are formatted in page headers, page footers, report headers, and report footers.

There are three different, broad types of values SQL*Plus can format: numbers, character strings, and dates. Not all commands can handle each type. With most commands, you can only specify number and date formats. The COLUMN command is a good example. The ACCEPT command is the only one that allows you to specify a date format string.

Format specification strings are made up of special characters that have meaning to SQL*Plus in the context of formatting a value for display. Numeric format strings, for example, tend to have lots of 0s, 9s, decimal points, and dollar signs.

Date format strings tend to include things like MM, DD, YYYY, and so forth. Character string formats are the simplest of all, because you basically have only one thing you can influence: length.

Formatting Numbers

SQL*Plus offers the most options when it comes to formatting numbers. Numeric format strings may contain any of the elements shown in Table B-1.

Table B-1. Numeric Format Elements

Format Element	Function
9	9s are used to control the number of significant digits to be displayed.
0	A 0 is used to mark the spot in the result where you want to begin displaying leading zeros. It replaces one of the 9s. The most common location for a 0 is at the extreme left of the format string, but you can place it elsewhere.
\$	Causes a number to be displayed with a leading dollar sign.
,	Places a comma in the output.
.	Marks the location of the decimal point.
B	Forces zero values to be displayed as blanks.
MI	Used at the end of a format string to cause a trailing negative sign to be displayed for negative values.
S	May be used at either the beginning or end of a format string, and causes a sign to be displayed. The + sign is used to mark positive numbers, and the - sign marks negative numbers. When you use S, a sign will always be displayed.
PR	Causes negative values to be displayed within angle brackets. For example, -123.99 will be displayed as "<123.99>". Positive values will be displayed with one leading and one trailing space in place of the angle brackets.
D	Marks the location of the decimal point.
G	Places a group separator (usually a comma) in the output.
C	Marks the place where you want the ISO currency indicator to appear. For US dollars, this will be USD.
L	Marks the place where you want the local currency indicator to appear. For US dollars, this will be the dollar sign character. You cannot use L and C in the same format specification.
V	Used to display scaled values. The number of digits to the right of the V indicates how many places to the right the decimal point is shifted before the number is displayed.
EEEE	Causes SQL*Plus to use scientific notation to display a value. You must use exactly four Es, and they must appear at the right end of the format string.

Table B-1. Numeric Format Elements (continued)

Format Element	Function
RN	Allows you to display a number using Roman numerals. This is the only format element where case makes a difference. An uppercase "RN" yields uppercase Roman numerals, while a lowercase "rn" yields Roman numerals in lowercase. Numbers displayed as Roman numerals must be integers, and must be between 1 and 3,999, inclusive.
DATE	Causes SQL*Plus to assume that the number represents a Julian date, and to display it in MM/DD/YY format.

To format a numeric column or other number, simply string together the format elements that yield the result you want. Except for the RN element, none of the numeric format elements are case-sensitive. Table B-2 contains a number of examples showing you how these format elements really work.

The ACCEPT command is unique in that it uses a format string to constrain the user's input. However, in doing so, it takes a rather loose interpretation of the format elements shown in Table B-1. You can read more about this in Chapter 7, *Advanced Scripting*. For the most part, though, only the 9, 0, and period are very useful with ACCEPT.

SQL*Plus always allows for a sign somewhere when you display a number. The default is for the sign to be positioned to the left of the number, and the sign is only displayed when the number is negative. Positive numbers will have a blank space in the leftmost position. Because space is always made for a sign character, number columns will typically be one space wider than your format specification seems to account for. That's the default behavior. Things change when you use S, MI, or PR. With S, you always get a sign. With MI, you get a trailing sign, or a trailing blank for positive numbers. PR gives you angle brackets, or spaces in place of them.

Table B-2. Numeric Format Examples

Value	Format	Result	Comments
123	9999	123	A basic number
1234.01	9,999.99	1,234.01	Comma and decimal point
23456	\$999,999.99	\$23,456.00	A dollar value
1	0999	0001	Leading zeros
1	99099	001	Leading zeros only within the rightmost three digits
23456	9,999.99	#####	An overflow condition
0	099B		Display zeros as blanks
1	099B	001	Leading zeros displayed, even with B, when the value is nonzero

Table B-2. Numeric Format Examples (continued)

Value	Format	Result	Comments
-1000.01	9,999.99mi	1,000.01-	Trailing minus sign
1000.01	9,999.99mi	1,000.01	Trailing space
-1001	S9,999	-1,001	Leading sign
-1001	9,999PR	<1,001>	Negative values in angle brackets
1001	9,999PR	1,001	Spaces instead of angle brackets
1001	9.999EEEE	-1.001E+03	Scientific notation
1995	RN	MCMXCV	Roman numerals, uppercase
1988	rn	mcmlxxxviii	Roman numerals, lowercase
1	date	01/01/12	Julian date, day one

Formatting Character Strings

SQL*Plus offers only one format element when it comes to character strings. That element is “A”. “A” is always followed by a number specifying the column width in characters. Character strings shorter than the column width are displayed left-justified within the column. Character strings that exceed the column width are either wrapped or truncated based on the option specified in the COLUMN command. The following example shows a text column that has been formatted wide enough to display the entire character string:

```
SQL> COLUMN a FORMAT A40
SQL> SELECT 'An apple a day keeps the doctor away.' A
       2 FROM dual;
```

```
A
-----
An apple a day keeps the doctor away.
```

You can format the column so that it is 18 characters wide, which results in the text being wrapped within that space:

```
SQL> COLUMN a FORMAT A18
SQL> SELECT 'An apple a day keeps the doctor away.' A
       2 FROM dual;
```

```
A
-----
An apple a day kee
ps the doctor away
```

By default, SQL*Plus wraps the text right in the middle of a word, if necessary. You can use the WORD_WRAPPED option of the COLUMN command to wrap text only at word boundaries.

```
SQL> COLUMN a FORMAT A18 WORD WRAPPED
SQL> SELECT 'An apple a day keeps the doctor away.' A
      2 FROM dual;
```

```
A
-----
An apple a day
keeps the doctor
away.
```

You also have the ability to truncate text at the column boundary.

```
SQL> COLUMN a FORMAT A18 TRUNCATE
SQL> SELECT 'An apple a day keeps the doctor away.' A
      2 FROM dual;
```

```
A
-----
An apple a day kee
```

When used with the ACCEPT command, a character format defines the maximum number of characters SQL*Plus will accept from the user. This is shown in the following example:

```
SQL> ACCEPT some_text CHAR FORMAT A10
thisthatthen
"thisthatthen" does not match input format "A10"

SQL>
```

While the character format used with ACCEPT specifies a maximum length, it does not specify a minimum length. You can always enter fewer characters than the format calls for, even to the point of entering nothing at all.

Formatting Dates

SQL*Plus doesn't really format dates at all. If you are selecting a date column from the database, you must use Oracle's built-in TO_CHAR function to convert the date to a character string, formatting it the way you want it. As far as SQL*Plus is concerned, that makes it just another character column. Table B-3 shows the date format elements that can be used with the TO_CHAR function.

The one SQL*Plus command that does recognize these date format elements is the ACCEPT command. When you ask the user to enter a date, you can also provide a date format specification. SQL*Plus will reject any date the user enters that does not match that format.

When displaying a date, you must use the TO_CHAR function to specify the format. The following example displays the current value of SYSDATE, including the time:

Table B-3. Date Format Elements

Format Element	Function
-/,,:;	Punctuation may be included anywhere in the date format string, and will be included in the output.
'text'	Quoted text may also be included in the date format string, and will be reproduced in the output.
AD or A.D. BC or B.C.	Includes an AD or BC indicator with the date.
AM or A.M. PM or P.M.	Prints AM or PM, whichever applies, given the time in question.
CC	The century number. This will be 20 for years 1900 through 1999.
SCC	Same as CC, but BC dates will be negative.
D	The number of the day of the week. This will be 1 through 7.
DAY	The name of the day. This will be Saturday, Sunday, Monday, and so forth.
DD	The day of the month.
DDD	The day of the year.
DY	The abbreviated name of the day. This will be Sat, Sun, Mon, and so forth.
HH	The hour of the day. This will be 1 through 12.
HH12	The hour of the day. This will be 1 through 12, the same as HH.
HH24	The hour of the day on a 24-hour clock. This will be 0–23.
IW	The week of the year. This will be 1–53.
IYYY	The four-digit year.
IYY	The last three digits of the year number.
IY	The last two digits of the year number.
I	The last digit of the year number.
J	The Julian day. Day 1 is equivalent to Jan 1, 4712 BC.
MI	The minute.
MM	The month number.
MON	The three-letter month abbreviation.
MONTH	The month name, fully spelled out.
Q	The quarter of the year. Quarter 1 is Jan–Mar, quarter 2 is Apr–Jun, and so forth.
RM	Is the month number in Roman numerals.
RR	When used with TO_CHAR, returns the last two digits of the year.
RRRR	When used with TO_CHAR, returns the four-digit year.
SS	The second.
SSSS	The number of seconds since midnight.
WW	The week of the year.

Table B-3. Date Format Elements (continued)

Format Element	Function
W	The week of the month. Week one starts on the first of the month. Week two starts on the 8th of the month, and so forth.
Y,YYY	The four-digit year with a comma after the first digit.
YEAR	The year spelled out in words.
SYEAR	The year spelled out in words, with a leading negative sign when the year is BC.
YYYY	The four-digit year.
SYYYYY	The four-digit year, with a leading negative sign when the year is BC.
YYY	The last three digits of the year number.
YY	The last two digits of the year number.
Y	The last digit of the year number.

```
SQL> SELECT TO_CHAR(SYSDATE, 'dd-Mon-yyyy hh:mi:ss PM')
2      FROM dual;
```

```
TO_CHAR(SYSDATE, 'DD-MON
-----
13-Dec-1998 09:13:59 PM
```

When you use a date format element that displays a text value, such as the name of a month, you need to pay attention to the case. The case of the element displayed will follow the case used when you specified the element. Suppose you want to display the three-letter abbreviation for a month. You could place either "Mon", "mon", or "MON" in your format string, and you would get back "Dec", "dec", or "DEC" respectively. You will see examples of this in Table B-4, which shows the results of several sample date format specifications.

To find out how to use a date format with the ACCEPT command, consult Chapter 7. ACCEPT uses the date format to validate what the user enters, and there are some limits on how closely the user is forced to follow that format.

Table B-4. Date Format Examples

Value	Format	Result
13-Dec-1998 09:13:59 PM	dd-mon-yyyy	13-dec-1998
13-Dec-1998 09:13:59 PM	dd-Mon-yyyy	13-Dec-1998
13-Dec-1998 09:13:59 PM	DD-MON-YYYY	13-DEC-1998
13-Dec-1998 09:13:59 PM	Month dd, yyyy	December 13, 1998
13-Dec-1998 09:13:59 PM	Month dd, yyyy "at" hh:mi am	December 13, 1998 at 09:13 pm
13-Dec-1998 09:13:59 PM	mm/dd/yy	12/13/98
13-Dec-1998 09:13:59 PM	mm/dd/rr	12/13/98

Table B-4. Date Format Examples (continued)

Value	Format	Result
13-Dec-1998 09:13:59 PM	mm/dd/yyyy	12/13/1998
13-Dec-1998 09:13:59 PM	Day	Sunday
13-Dec-1998 09:13:59 PM	ddd	347
13-Dec-1998 09:13:59 PM	ww	50
13-Dec-1998 09:13:59 PM	q	4
13-Dec-1998 09:13:59 PM	year	nineteen ninety-eight
13-Dec-1998 09:13:59 PM	Year	Nineteen Ninety-Eight
13-Dec-1998 09:13:59 PM	YEAR	NINETEEN NINETY-EIGHT

Oracle SQL*Plus: The Definitive Guide



SQL*Plus is an interactive query tool that's available at every Oracle site, from the largest data warehouse to the smallest single-user system. The tool continues to be supported and enhanced with each new version of Oracle, including Oracle8 and Oracle8i. Despite the wide availability and use of SQL*Plus, however, few Oracle developers and DBAs know how powerful a tool SQL*Plus can be.

This book is truly the definitive guide to SQL*Plus. It describes SQL*Plus from start to finish. It provides a quick reference to all of the tool's commands and options. Most importantly, it describes, in step-by-step fashion, how to perform all of the tasks you could possibly want to perform (and many you might not realize you *could* perform) with SQL*Plus. For example, you'll learn how to:

- Write and execute script files
- Generate and format ad-hoc reports
- Extract data from the database
- Query the data dictionary tables
- Customize your SQL*Plus environment
- Use the SQL*Plus administrative features (new in Oracle8i; these features previously required Server Manager)

*Oracle SQL*Plus: The Definitive Guide* is an indispensable resource for readers who are new to SQL*Plus, a task-oriented learning tool for those who are already using it, and a quick reference for every user.

*"This book is a great companion to Oracle's documentation. Jonathan offers practical advice and assistance to help you get the most out of SQL*Plus. I learned a few things from this book!"*

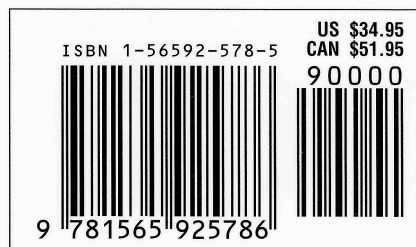
—Alison Holloway, Product and Documentation Manager
Australian Product Development Centre, Oracle Corporation

*"In working with Oracle since version 5, I have found that SQL*Plus is indispensable as the standard interface linking into the heart of the Oracle RDBMS. Its nature as the universal tool for all versions of Oracle places it on the must-know list for every developer and database administrator. And Jonathan's book is the preeminent quick-start and tips reference for all levels of Oracle expertise."*

—David Kaufman, Oracle Certified Professional, TUSC

"I enjoyed the conversational style of the book; it almost felt like a personal interaction with the author. Certainly a relaxed, yet very thorough, presentation of material. The term "Definitive Guide" is very appropriate."

—John-Paul Navarro, Senior Support Analyst, Argonne National Laboratory



Visit O'Reilly on the Web at www.oreilly.com

