SQL*PLUS 命令的使用大全

Oracle 的 sql*plus 是与 oracle 进行交互的客户端工具。在 sql*plus 中，可以运行 sql*plus 命令与 sql*plus 语句。

我们通常所说的 DML、DDL、DCL 语句都是 sql*plus 语句，它们执行完后，都可以保存在一个被称为 sql buffer 的内存区域中，并且只能保存一条最近执行的 sql 语句，我们可以对保存在 sql buffer 中的 sql 语句进行修改，然后再次执行，sql*plus 一般都与数据库打交道。

除了 sql*plus 语句，在 sql*plus 中执行的其它语句我们称之为 sql*plus 命令。它们执行完后，不保存在 sql buffer 的内存区域中，它们一般用来对输出的结果进行格式化显示，以便于制作报表。

下面就介绍一下一些常用的 sql*plus 命令：

1. 执行一个SQL脚本文件

SQL>start file_name
SQL>@ file_name
我们可以将多条 sql 语句保存在一个文本文件中，这样当要执行这个文件中的所有的 sql 语句时，用上面的任一命令即可，这类似于 dos 中的批处理。

@与@@的区别是什么？
@等于 start 命令，用来运行一个 sql 脚本文件。
@命令**调用当前目录下的，或指定全路径，或可以通过 SQLPATH 环境变量搜寻到的脚本文件**。该命令使用是一般要指定要执行的文件的全路径，否则从缺省路径(可用 SQLPATH 变量指定)下读取指定的文件。
@@用在 sql 脚本文件中，用来说明用@@执行的 sql 脚本文件与@@所在的文件在同一目录下，**而不用指定要执行 sql 脚本文件的全路径，也不是从 SQLPATH 环境变量指定的路径中寻找 sql 脚本文件**，该命令一般用在脚本文件中。
如：在 c:\temp 目录下有文件 start.sql 和 nest_start.sql，start.sql 脚本文件的内容为：
@@nest_start.sql　　- - 相当于@ c:\temp\nest_start.sql
则我们在 sql*plus 中，这样执行：
SQL> @ c:\temp\start.sql

2. 对当前的输入进行编辑

SQL>edit

3. 重新运行上一次运行的sql语句

SQL>/

4. 将显示的内容输出到指定文件

SQL> SPOOL file_name
在屏幕上的所有内容都包含在该文件中，包括你输入的 sql 语句。

5. 关闭spool输出

SQL> SPOOL OFF
只有关闭 spool 输出，才会在输出文件中看到输出的内容。

6．显示一个表的结构

SQL> desc table_name

7. COL命令：

主要格式化列的显示形式。
该命令有许多选项，具体如下：
COL[UMN] [{ column|expr} [option ...]]
Option 选项可以是如下的子句:
ALI[AS] alias
CLE[AR]
FOLD_A[FTER]
FOLD_B[EFORE]
FOR[MAT] format
HEA[DING] text
JUS[TIFY] {L[EFT]|C[ENTER]|C[ENTRE]|R[IGHT]}
LIKE { expr|alias}
NEWL[INE]
NEW_V[ALUE] variable
NOPRI[NT]|PRI[NT]
NUL[L] text
OLD_V[ALUE] variable
ON|OFF
WRA[PPED]|WOR[D_WRAPPED]|TRU[NCATED]

1)．改变缺省的列标题

COLUMN column_name HEADING column_heading
For example:

```
Sql>select * from dept;
        DEPTNO DNAME                                LOC
---------- -------------------------- ---------
            10 ACCOUNTING                       NEW YORK
sql>col    LOC heading location
sql>select * from dept;
       DEPTNO DNAME                                location
--------- -------------------------- -----------
            10 ACCOUNTING                       NEW YORK
```

2)．将列名ENAME改为新列名EMPLOYEE NAME并将新列名放在两行上：

Sql>select * from emp

```
Department    name                    Salary
---------- ---------- ----------
            10 aaa                        11
SQL> COLUMN ENAME HEADING 'Employee|Name'
Sql>select * from emp
              Employee
Department    Name                    Salary
---------- ---------- ----------
            10 aaa                        11
```

note: the col heading turn into two lines from one line.

3)．改变列的显示长度：

```
FOR[MAT] format
Sql>select empno,ename,job from emp;
        EMPNO ENAME          JOB
---------- ----------        ---------
          7369 SMITH          CLERK
          7499 ALLEN          SALESMAN
7521 WARD           SALESMAN
Sql> col ename format a40
        EMPNO ENAME                                              JOB
---------- ----------------------------------------              ---------
          7369 SMITH                                              CLERK
          7499 ALLEN                                              SALESMAN
          7521 WARD                                               SALESMAN
```

4)．设置列标题的对齐方式

```
JUS[TIFY] {L[EFT]|C[ENTER]|C[ENTRE]|R[IGHT]}
SQL> col ename justify center
SQL> /
        EMPNO                ENAME                          JOB
---------- ----------------------------------------          ---------
          7369 SMITH                                          CLERK
          7499 ALLEN                                          SALESMAN
7521 WARD                                                  SALESMAN
```

对于 NUMBER 型的列，列标题缺省在右边，其它类型的列标题缺省在左边

5)．不让一个列显示在屏幕上

```
NOPRI[NT]|PRI[NT]
SQL> col job noprint
SQL> /
        EMPNO                ENAME
---------- ----------------------------------------
          7369 SMITH
          7499 ALLEN
7521 WARD
```

6). 格式化NUMBER类型列的显示：

```
SQL> COLUMN SAL FORMAT $99,990
SQL> /
Employee
Department Name              Salary       Commission
---------- ---------- --------- ----------
30                ALLEN             $1,600       300
```

7). 显示列值时，如果列值为NULL值，用text值代替NULL值

```
COMM NUL[L] text
SQL>COL COMM NUL[L] text
```

8). 设置一个列的回绕方式

```
WRA[PPED]|WOR[D_WRAPPED]|TRU[NCATED]
          COL1
--------------------
HOW ARE YOU?

SQL>COL COL1 FORMAT A5
SQL>COL COL1 WRAPPED
COL1
-----
HOW A
RE YO
U?

SQL> COL COL1 WORD_WRAPPED
COL1
-----
HOW
ARE
YOU?

SQL> COL COL1 TRUNCATED
COL1
-----
HOW A
```

9). 显示列的当前的显示属性值

```
SQL> COLUMN column_name
```

10). 将所有列的显示属性设为缺省值

```
SQL> CLEAR COLUMNS
```

8. 屏蔽掉一个列中显示的相同的值

BREAK ON break_column
SQL> BREAK ON DEPTNO
SQL> SELECT DEPTNO, ENAME, SAL
FROM EMP
WHERE SAL < 2500
ORDER BY DEPTNO;
DEPTNO ENAME SAL
---------- ---------- ---------
10 CLARK 2450
MILLER 1300
20 SMITH 800
ADAMS 1100

9. 在上面屏蔽掉一个列中显示的相同的值的显示中，每当列值变化时在值变化之前插入n个空行。

BREAK ON break_column SKIP n

SQL> BREAK ON DEPTNO SKIP 1
SQL> /
DEPTNO ENAME SAL
---------- ---------- ---------
10 CLARK 2450
MILLER 1300

20 SMITH 800
ADAMS 1100

10. 显示对BREAK的设置

SQL> BREAK

11. 删除 8、9 的设置

SQL> CLEAR BREAKS

12. Set 命令：

该命令包含许多子命令：
SET system_variable value
system_variable value 可以是如下的子句之一：
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] [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]}]
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}
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}

1).设置当前session是否对修改的数据进行自动提交

SQL>SET AUTO[COMMIT] {ON|OFF|IMM[EDIATE]| n}

2).在用start命令执行一个sql脚本时，是否显示脚本中正在执行的SQL语句

SQL> SET ECHO {ON|OFF}

3).是否显示当前sql语句查询或修改的行数

SQL> SET FEED[BACK] {6|n|ON|OFF}
默认只有结果大于 6 行时才显示结果的行数。如果 set feedback 1 ，则不管查询到多少行都返回。当为 off 时，一律不显示查询的行数

4).是否显示列标题

SQL> SET HEA[DING] {ON|OFF}
当 set heading off 时，在每页的上面不显示列标题，而是以空白行代替

5).设置一行可以容纳的字符数

SQL> SET LIN[ESIZE] {80|n}
如果一行的输出内容大于设置的一行可容纳的字符数，则折行显示。

6).设置页与页之间的分隔

SQL> SET NEWP[AGE] {1|n|NONE}
当 set newpage 0 时，会在每页的开头有一个小的黑方框。
当 set newpage n 时，会在页和页之间隔着 n 个空行。
当 set newpage none 时，会在页和页之间没有任何间隔。

7).显示时，用text值代替NULL值

SQL> SET NULL text

8).设置一页有多少行数

SQL> SET PAGES[IZE] {24|n}
如果设为 0，则所有的输出内容为一页并且不显示列标题

9).是否显示用DBMS_OUTPUT.PUT_LINE包进行输出的信息。

SQL> SET SERVEROUT[PUT] {ON|OFF}
在编写存储过程时，我们有时会用 dbms_output.put_line 将必要的信息输出，以便对存储过程进行调试，只有将 serveroutput 变量设为 on 后，信息才能显示在屏幕上。

10).当SQL语句的长度大于LINESIZE时，是否在显示时截取SQL语句。

SQL> SET WRA[P] {ON|OFF}
当输出的行的长度大于设置的行的长度时（用 set linesize n 命令设置），当 set wrap on 时，输出行的多于的字符会另起一行显示，否则，会将输出行的多于字符切除，不予显示。

11).是否在屏幕上显示输出的内容，主要用与SPOOL结合使用。

SQL> SET TERM[OUT] {ON|OFF}
在用 spool 命令将一个大表中的内容输出到一个文件中时，将内容输出在屏幕上会耗费大量的时间，设置 set termout off 后，则输出的内容只会保存在输出文件中，不会显示在屏幕上，极大的提高了 spool 的速度。

12).将SPOOL输出中每行后面多余的空格去掉

SQL> SET TRIMS[POOL] {ON|OFF}

13)显示每个sql语句花费的执行时间

set TIMING {ON|OFF}

14). 遇到空行时不认为语句已经结束，从后续行接着读入。

SET SQLBLANKLINES ON
Sql*plus 中, 不允许 sql 语句中间有空行, 这在从其它地方拷贝脚本到 sql*plus 中执行时很麻烦. 比如下面的脚本:
select deptno, empno, ename
from emp

where empno = '7788';
如果拷贝到 sql*plus 中执行, 就会出现错误。这个命令可以解决该问题

15).设置DBMS_OUTPUT的输出

SET SERVEROUTPUT ON BUFFER 20000
用 dbms_output.put_line('strin_content');可以在存储过程中输出信息，对存储过程进行调试
如果想让 dbms_output.put_line(' abc');的输出显示为：
SQL> abc，而不是 SQL>abc，则在 SET SERVEROUTPUT ON 后加 format wrapped 参

数。

16). 输出的数据为html格式

set markup html
在 8.1.7 版本(也许是 816? 不太确定)以后, sql*plus 中有一个 set markup html 的命令, 可以将 sql*plus 的输出以 html 格式展现.
注意其中的 spool on, 当在屏幕上输出的时候, 我们看不出与不加 spool on 有什么区别, 但是当我们使用 spool filename 输出到文件的时候, 会看到 spool 文件中出现了等 tag.

14. 修改sql buffer中的当前行中，第一个出现的字符串

```
C[HANGE] /old_value/new_value
SQL> l
  1* select * from dept
SQL> c/dept/emp
  1* select * from emp
```

15. 编辑sql buffer中的sql语句

EDI[T]

16. 显示sql buffer中的sql语句，list n显示sql buffer中的第n行，并使第n行成为当前行

L[IST] [n]

17. 在sql buffer的当前行下面加一行或多行

I[NPUT]

18. 将指定的文本加到sql buffer的当前行后面

```
A[PPEND]
SQL> select deptno,
  2  dname
  3  from dept;
    DEPTNO DNAME
---------- --------------
        10 ACCOUNTING
        20 RESEARCH
        30 SALES
        40 OPERATIONS
```

```
SQL> L 2
  2* dname
SQL> a ,loc
  2* dname,loc
SQL> L
  1  select deptno,
  2  dname,loc
  3* from dept
SQL> /

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON
```

19．将sql buffer中的sql语句保存到一个文件中

SAVE file_name

20．将一个文件中的sql语句导入到sql buffer中

GET file_name

21．再次执行刚才已经执行的sql语句

RUN
or
/

22．执行一个存储过程

EXECUTE procedure_name

23．在sql*plus中连接到指定的数据库

CONNECT user_name/passwd@db_alias

24．设置每个报表的顶部标题

TTITLE

25．设置每个报表的尾部标题

BTITLE

26．写一个注释

REMARK [text]

27．将指定的信息或一个空行输出到屏幕上

PROMPT [text]

28．将执行的过程暂停，等待用户响应后继续执行

PAUSE [text]

Sql>PAUSE Adjust paper and press RETURN to continue.

29．将一个数据库中的一些数据拷贝到另外一个数据库（如将一个表的数据拷贝到另一个数据库）

COPY {FROM database | TO database | FROM database TO database}
{APPEND|CREATE|INSERT|REPLACE} destination_table
[(column, column, column, ...)] USING query

sql>COPY FROM SCOTT/TIGER@HQ TO JOHN/CHROME@WEST
create emp_temp
USING SELECT * FROM EMP

30．不退出sql*plus，在sql*plus中执行一个操作系统命令：

HOST

Sql> host hostname
该命令在 windows 下可能被支持。

31．在sql*plus中，切换到操作系统命令提示符下，运行操作系统命令后，可以再次切换回sql*plus：

!

sql>!

$hostname
$exit
sql>

该命令在 windows 下不被支持。

32．显示sql*plus命令的帮助

HELP
如何安装帮助文件：
Sql>@ ?\sqlplus\admin\help\hlpbld.sql ?\sqlplus\admin\help\helpus.sql
Sql>help index

33．显示sql*plus系统变量的值或sql*plus环境变量的值

Syntax
SHO[W] option
where option represents one of the following terms or clauses:
system_variable
ALL
BTI[TLE]
ERR[ORS] [{FUNCTION|PROCEDURE|PACKAGE|PACKAGE BODY|
TRIGGER|VIEW|TYPE|TYPE BODY} [schema.]name]
LNO
PARAMETERS [parameter_name]
PNO
REL[EASE]
REPF[OOTER]
REPH[EADER]
SGA
SPOO[L]
SQLCODE
TTI[TLE]
USER

1）．显示当前环境变量的值：

Show all

2）．显示当前在创建函数、存储过程、触发器、包等对象的错误信息

Show error
当创建一个函数、存储过程等出错时，变可以用该命令查看在那个地方出错及相应的出错信息，进行修改后再次进行编译。

3）．显示初始化参数的值：

show PARAMETERS [parameter_name]

4）．显示数据库的版本：

show REL[EASE]

5）．显示SGA的大小

show SGA

6）．显示当前的用户名

show user

34.查询一个用户下的对象

SQL>select * from tab;
SQL>select * from user_objects;

35.查询一个用户下的所有的表

SQL>select * from user_tables;

36.查询一个用户下的所有的索引

SQL>select * from user_indexes;

37. 定义一个用户变量

方法有两个：
a. define
b. COL[UMN] [{column|expr} NEW_V[ALUE] variable [NOPRI[NT]|PRI[NT]]
OLD_V[ALUE] variable [NOPRI[NT]|PRI[NT]]

下面对每种方式给予解释：
a. Syntax
DEF[INE] [variable]|[variable = text]
定义一个用户变量并且可以分配给它一个 CHAR 值。

assign the value MANAGER to the variable POS, type:
SQL> DEFINE POS = MANAGER

assign the CHAR value 20 to the variable DEPTNO, type:
SQL> DEFINE DEPTNO = 20

list the definition of DEPTNO, enter
SQL> DEFINE DEPTNO
————————————————————
DEFINE DEPTNO = ”20” (CHAR)

定义了用户变量 POS 后，就可以在 sql*plus 中用&POS 或&&POS 来引用该变量的值，
sql*plus 不会再提示你给变量输入值。

b. COL[UMN] [{column|expr} NEW_V[ALUE] variable [NOPRI[NT]|PRI[NT]]
NEW_V[ALUE] variable
指定一个变量容纳查询出的列值。
例:column col_name new_value var_name noprint
select col_name from table_name where ……..
将下面查询出的 col_name 列的值赋给 var_name 变量.

一个综合的例子:
得到一个列值的两次查询之差(此例为 10 秒之内共提交了多少事务):
column redo_writes new_value commit_count

select sum(stat.value) redo_writes
from v$sesstat stat, v$statname sn
where stat.statistic# = sn.statistic#
and sn.name = 'user commits';

-- 等待一会儿(此处为 10 秒);
execute dbms_lock.sleep(10);

set veri off
select sum(stat.value) - &commit_count commits_added
from v$sesstat stat, v$statname sn
where stat.statistic# = sn.statistic#
and sn.name = 'user commits';

38. 定义一个绑定变量

VAR[IABLE] [variable [NUMBER|CHAR|CHAR (n)|NCHAR|NCHAR (n) |VARCHAR2 (n)|NVARCHAR2 (n)|CLOB|NCLOB|REFCURSOR]]
定义一个绑定变量，该变量可以在 pl/sql 中引用。
可以用 print 命令显示该绑定变量的信息。
如：
column inst_num heading "Inst Num" new_value inst_num format 99999;
column inst_name heading "Instance" new_value inst_name format a12;
column db_name heading "DB Name" new_value db_name format a12;
column dbid heading "DB Id" new_value dbid format 9999999999 just c;

prompt
prompt Current Instance

```
prompt ~~~~~~~~~~~~~~~~~

select d.dbid                    dbid
       , d.name                    db_name
       , i.instance_number inst_num
       , i.instance_name      inst_name
  from v$database d,
          v$instance i;

variable dbid            number;
variable inst_num      number;
begin
   :dbid          :=   &dbid;
   :inst_num    :=   &inst_num;
end;
/
```

说明：
在 sql*plus 中，该绑定变量可以作为一个存储过程的参数，也可以在匿名 PL/SQL 块中直接引用。为了显示用 VARIABLE 命令创建的绑定变量的值，可以用 print 命令

注意：
绑定变量不同于变量：
1. 定义方法不同
2. 引用方法不同
绑定变量：:variable_name
变量：&variable_name or &&variable_name
3.在 sql*plus 中，可以定义同名的绑定变量与用户变量，但是引用的方法不同。

39. &与&&的区别

&用来创建一个临时变量，每当遇到这个临时变量时，都会提示你输入一个值。
&&用来创建一个持久变量，就像用用 define 命令或带 new_vlaue 字句的 column 命令创建的持久变量一样。当用&&命令引用这个变量时，不会每次遇到该变量就提示用户键入值，而只是在第一次遇到时提示一次。

如，将下面三行语句存为一个脚本文件，运行该脚本文件，会提示三次，让输入 deptnoval 的值：

```
select count(*) from emp where deptno = &deptnoval;
select count(*) from emp where deptno = &deptnoval;
select count(*) from emp where deptno = &deptnoval;
```

将下面三行语句存为一个脚本文件，运行该脚本文件，则只会提示一次，让输入 deptnoval 的值：

```
select count(*) from emp where deptno = &&deptnoval;
select count(*) from emp where deptno = &&deptnoval;
select count(*) from emp where deptno = &&deptnoval;
```

40．在输入sql语句的过程中临时先运行一个sql*plus命令

#
有没有过这样的经历？在 sql*plus 中敲了很长的命令后，突然发现想不起某个列的名字了，

如果取消当前的命令,待查询后再重敲, 那太痛苦了. 当然你可以另开一个 sql*plus 窗口进行查询, 但这里提供的方法更简单.

比如说, 你想查工资大于 4000 的员工的信息, 输入了下面的语句:

SQL> select deptno, empno, ename
2 from emp
3 where
这时, 你发现你想不起来工资的列名是什么了.

这种情况下, 只要在下一行以#开头, 就可以执行一条 sql*plus 命令, 执行完后, 刚才的语句可以继续输入

SQL>> select deptno, empno, ename
2 from emp
3 where
6 #desc emp
Name Null? Type
--- -------- --------------
EMPNO NOT NULL NUMBER(4)
ENAME VARCHAR2(10)
JOB VARCHAR2(9)
MGR NUMBER(4)
HIREDATE DATE
SAL NUMBER(7,2)
COMM NUMBER(7,2)
DEPTNO NUMBER(2)

6 sal > 4000;

DEPTNO EMPNO ENAME
---------- ---------- ----------
10 7839 KING

41. SQLPlus中的快速复制和粘贴技巧

1) 鼠标移至想要复制内容的开始
2) 用右手食指按下鼠标左键
3) 向想要复制内容的另一角拖动鼠标，与 Word 中选取内容的方法一样
4) 内容选取完毕后（所选内容全部反显），鼠标左键按住不动，用右手中指按鼠标右键
5) 这时，所选内容会自动复制到 SQL*Plus 环境的最后一行