oracle 10g使用SQL Plus

1. 编辑SQL语句

A[PPEND] text : 将text附加到当前行之后

C[HANGE]/old/new : 将当前行中的old替换为new

CL[EAR] BUFF[ER] : 清楚缓冲区中的所有行

DEL : 删除当前行

DEL x : 删除第x行（行号从1开始）

L[IST] : 列出缓冲区的所有行

L[IST] x : 列出缓冲区的第x行

R[UN] : 运行缓冲区中保存的语句

/ : 运行缓冲区中保存的语句

x : 将第x行作为当前行

2. 保存、检索并运行文件

SAV[E] filename [{ REPLACE | APPEND }] : 将SQL*Plus缓冲区的内容保存到由filename所指定的文件中.APPEND选项指定将缓冲区的内容追加到一个现有文件之后，REPLACE选项指定覆盖一个现有文件。

GET filename : 将filename所指定的文件的内容读入SQL*Plus缓冲区中。

STA[RT] filename : 将由filename所指定的文件的内容读入SQL*Plus缓冲区中，并运行缓冲区中的内容。

@filename : 同上START filename

ED[IT] : 将缓冲区的内容复制到一个名为afiedt.buf的文件中，然后启动操作系统默认的编辑器。在退出编辑器是，所编辑的文件将被复制到SQL*Plus缓冲区中

ED[IT] filename : 指定要编辑的文件，

SPO[OL] filename : 将SQL*Plus中的输出结果复制到filename所指定的文件中。

SPO[OL] OFF : 将SQL*Plus中的输出结果复制到filename所指定的文件中，并关闭该文件。

3. 格式化列

COL[UMN] { column | alias} [options]

column指定列明

alias指定要格式化的列的别名

options指定用于格式化列或别名的一个或多个选项。

FOR[MAT] format : 将列或别名的显示格式设置为有format字符串指定的格式

HEA[DING] heading : 将列或别名的标题中的文本设置为由heading字符串指定的格式

JUS[TIFY] [{ left | center | right }]: 将列输出设置为左对齐、居中或右对齐

WRA[PPED] : 在输出结果中将一个字符串的末尾换行显示。该选项可能导致单个单词跨越多行。

WOR[D_WRAPPED] : 与WRAPPED选项类似，不同支出在于单个单词不能跨行。

CLE[AR] : 清楚列的任何格式化（将格式设置回默认值）

4. 设置页面大小

SET PAGESIZE 100

5. 设置行大小

SET LINESIZE 50

6. 清楚列格式

COLUMN product_id CLEAR

7. 使用变量

临时变量 ： 只在使用它的SQL语句中有效，值不能保留

已定义变量 ： 值会一直保留到被显示地删除、重定义或退出SQL*Plus为止。

1） 临时变量 &定义临时变量

eg :

```
select product_id, name, price from products where product_id = &product_id_var;
```

1. 控制输出行 ：

set verify off : 禁止显示原行和新行

set vefify on : 开启显示原行和新行

2. 修改变量定义字符

SET DEFINE 命令用于指定一个除字符 & 之外的字符，用来定义变量。

eg : 使用SET DEFINE命令将变量定义字符设置为井号字符（#）

```
SET DEFINE '#'
select product_id, name, price from products  where product_id = #product_id_var;
```

3. 使用变量替换表名和列名

eg :

```
select name, &col_var, from &table_var where &col_var = &col_val;
```

note : &&用于防止重复输入一个变量

eg :

```
select name, &&col_var from &table_name where &&col_var = &col_val;
```

2） 已定义变量

1. 使用DEFINE命令定义变量，使用UNDEFINE命令删除变量

eg :

DEFINE product_id_var = 7 ： 定义变量product_id_var, 并为其赋值为7

DEFINE product_id_var : 显示product_id_var的定义

DEFINE : 显示当前会话的所有变量

eg :

```
select product_id, name, price from products where product_id = &product_id_var;
```

2. 使用ACCEPT命令定义并设置变量

ACCEPT命令用于等待用户为变量输入一个值。

ACCEPT variable_name [type] [FORMAT format] [PROMPT prompt] [HIDE]

prompt 指定了SQL*Plus所显示的提示文本，提示用户输入变量的值

HIDE 说明隐藏为变量输入的值（用*号显示）

eg :

```
ACCEPT customer_id_var NUMBER FORMAT 99 PROMPT 'Customer id: '
ACCEPT date_var DATE FORMAT 'DD-MON-YYYY' PROMPT 'Date: '
ACCEPT password_var CHAR PROMPT 'Password: ' HIDE
```

3. 使用UNDEFINE命令删除变量

```
UNDEFINE customer_id_var
UNDEFINE date_var
UNDEFINE password_var
```

8. 创建简单报表

1）在脚本中使用临时变量

vi report1.sql

```
-- suppress display of the statements and verification messages
SET ECHO OFF
SET VERIFY OFF
select product_id, name, price from products where product_id = &product_id_var;
@report1.sql
```

2）在脚本中使用已定义变量

vi report2.sql

```
SET ECHO OFF
SET VERIFY OFF
ACCEPT product_id_var, NUMBER FORMAT 99 PROMPT 'Enter product id: '
select product_id, name, price from products where product_id = &product_id_var;
-- clean up
UNDEFINE product_id_var
@report2.sql
```

3）向脚本中的变量传递值

在运行脚本时，可以想变量传递值，这样做时，必须使用一个数字来引用脚本中的变量。

vi report3.sql

```
SET ECHO OFF
SET VERIFY OFF
select product_id, name, price from products where product_id = &1;
@report3.sql 3
```

执行脚本时可以添加任意个参数，命令行中指定的每个值都会对应于文件中的一个匹配数字。

第一个参数对应&1， 第二个参数对应&2，以此类推。

eg :

```
vi report4.sql
SET ECHO OFF
SET VERIFY OFF
select product_id, product_type_id, name, price from products where product_type_id = &1 and price > &2;
@report4.sql 1 9.99
```

4）添加页眉和页脚

eg :

```
vi report5.sql
TTITLE 'Product Report'
BTITLE 'Thanks for running the report'
SET ECHO OFF
SET VERIFY OFF
SET PAGESIZE 30
SET LINESIZE 70
CLEAR COLUMNS
COLUMN product_id, HEADING_ID FORMAT 99
COLUMN name HEADING 'Product Name' FORMAT A20 WORD_WRAPPED
COLUMN description HEADING Description FORMAT A30 WORD_WRAPPED
COLUMN price HEADING Price FORMAT $99.99
select product_id, name, description, price from products;
CLEAR COLUMNS
@report5.sql
```

5）计算小计

BREAK ON 和COMPUTE命令可以结合使用，用来为列添加小计。

BREAK ON子句可以让SQL*Plus根据劣质的范围分隔输出结果

COMPUTE子句可以让SQL*Plus计算一列的和。

eg : 对相同类型的产品进行小计

```
vi report6.sql
BREAK ON product_type_id
COMPUTE SUM OF price ON product_type_id
SET ECHO OFF
SET VERIFY OFF
SET PAGESIZE 50
SET LINESIZE 70
CLEAR COLUMNS
COLUMN price HEADING Price FORMAT $999/99
select product_type_id, name, price from products order by product_type_id;
CLEAR COLUMNS
@report6.sql
```

9. 自动生成SQL语句

eg :

```
select 'DROP TABLE ' || table_name || ';' from user_tables;
```