L'outil SQL*PLUS

Table des matières

1 Première utilisation

L'accès et l'utilisation d'une base de données ORACLE peut se faire à l'aide de différents outils.
L'outil historique pour l'utilisation du SQL et du PL/SQL est l'interface utilisateur SQL*Plus.
Ce programme permet aux utilisateurs finaux, aux développeurs et aux administrateurs les fonctionnalités suivantes :

- **manipulation et exécution de commandes SQL et de blocs PL/SQL.**
- **mise en forme des résultats de requêtes.**
- **visualisation des structures des tables et copie de données inter-base.**
- **commandes et opérations d'entrée/sortie (saisie, affichage, manipulation de variables).**

Rq: SQL*Plus possède également ses propres ordres de programmation. Il faut veiller à ne pas confondre l'utilitaire SQL*Plus et le langage SQL.
Avant la version 11, l'outil SQL*Plus se déclinait sous trois formes différentes :

- **outil ligne de commande (sqlplus.exe) ;**
- **application graphique Windows (sqlplusw.exe) ;**
- **application Web (iSQL*Plus).**

Depuis la version 11, seule la version ligne de commande subsiste.

1.1 Connexion et déconnexion

Pour pouvoir utiliser le SQL, il faut se connecter à la base de données, c'est-à-dire fournir un nom d'utilisateur, éventuellement protégé par un mot de passe.
SQL*Plus permet, soit de se connecter automatiquement en passant le nom et le mot de passe en paramètres de la ligne de commande du système d'exploitation, soit de demander le nom ou le mot de passe après le lancement.
On peut, en outre, changer de nom d'utilisateur en cours de session SQL*Plus en se reconnectant.

1.1.1 Lancement du programme

À partir du prompt du système d'exploitation ;

Syntaxe

```
sqlplus[[-S] [nomuser[/motpasse][@chaîne_de_connexion]]
```

avec :

`-S`

Mode silencieux.

`nomuser`

Nom de connexion à la base.

`motpasse`

Mot de passe de l'utilisateur.

`chaîne_de_connexion`

Nom du service défini dans le fichier TNSNAMES.ORA à utiliser pour se connecter au serveur Oracle.

Rq: Les différents ordres permettant la configuration de SQL*Plus sont détaillés par la suite.

Si l'on souhaite retrouver toujours le même environnement de travail, il est possible de stocker ses préférences dans le ficher login.sql. Le fichier login.sql du répertoire courant est exécuté à la première connexion à SQL*Plus.

Exemples

Lancement du programme avec passage en paramètres du nom de l'utilisateur (TOTO) et du mot de passe (PIZZA) :

```
$sqlplus toto/pizza

Connecté à :

...........

SQL>
```

Lancement du programme avec demande de connexion (le mot de passe est saisi en frappe aveugle) :

```
$sqlplus

username : toto

password :

Connecté à :

.............

SQL>
```

1.1.2 Connexion après lancement

CONNECT

Syntaxe

```
CONNECT [nomuser[/motpasse]][@chaîne_de_connexion]]
```

Exemple

L'utilisateur courant veut se connecter sous le nom FLORIAN :

```
SQL> Connect florian

password : < Saisie du mot de passe en frappe aveugle>

connecté.

SQL>
```

La chaîne de connexion n'est nécessaire que si l'on souhaite se connecter à un serveur Oracle qui n'est pas local. On peut donc se passer de chaîne de connexion lorsque l'on exécute SQL*PLUS directement sur le serveur Oracle.

Connexion à une base distante depuis SQL*PLUS

```
SQL> connect scott/tiger@dbhelios
Connecté.
SQL>
```

1.1.3 Changement du mot de passe

Permet à l'utilisateur de changer son mot de passe dans l'environnement SQL*Plus.

PASSWORD

Syntaxe

```
PASSWORD [nomuser]
```

Exemple

Changement du mot de passe de l'utilisateur courant.

```
SQL> PASSWORD
Modification de mot de passe pour LIVRE
Ancien mot de passe : ***
Nouveau mot de passe : ***
Entrer le nouveau mot de passe : ***
SQL>
```

1.1.4 Déconnexion

Après déconnexion, l'utilisateur ne peut plus utiliser de commande SQL ou PL/SQL.

DISCONNECT

Syntaxe

```
DISCONNECT
```

Exemple

Déconnexion sans sortie de SQL*Plus :

```
SQL>DISCONNECT

Disconnected from .......

.......

SQL>
```

1.1.5 Sortie de SQL*Plus

EXIT

La sortie du programme entraîne la déconnexion à la base.

Syntaxe

```
EXIT [SUCCESS / FAILURE / WARNING /n /variable
/:variable_liée][COMMIT / ROLLBACK]

SUCCESS/FAILURE/WARNING/n/ variable/:variable_liée
```

Permet de communiquer au système d'exploitation un code de retour sur l'exécution de la session.

```
COMMIT/ROLLBACK
```

Valide ou annule la transaction en cours au moment de la fin de session (COMMIT par défaut).

Exemple

Sortie de SQL*Plus et retour au système d'exploitation ($: prompt UNIX).

```
SQL> EXIT

$.
```

1.2 Exécution des instructions

Après la connexion, l'utilisateur peut utiliser SQL*Plus pour saisir des commandes SQL, des blocs PL/SQL ou des commandes SQL*Plus, à partir du prompt.
Les commandes SQL et PL/SQL peuvent être saisies sur plusieurs lignes (on marque une fin de ligne par la touche [Entrée]), le caractère de fin de commande étant le point-virgule (;). Pour améliorer la lisibilité on peut insérer dans la syntaxe autant d'espaces ou de tabulations qu'on le souhaite. Il n'y a pas de distinction entre majuscules et minuscules (sauf utilisation de guillemets).

Exemple

Création de lignes dans la table CLIENTS :

```
SQL> insert into CLIENTS (nocli, nom) values (10, 'TOTO') ;

1 ligne creee

SQL> insert into

2> Clients (Nocli, Nom)

3> values (15, 'Titi')

4> ;

1 ligne creee

SQL>
```

Les commandes SQL*Plus ne nécessitent pas de caractère de fin de commande, la marque de fin de ligne ([Entrée]) suffit. Si une commande doit être saisie sur plusieurs lignes, il faut utiliser le caractère de césure de ligne (tiret (-)).

Exemple

Déclaration de format de colonne :

```
SQL> column REFART heading "Référence"

SQL> column DESIGN -

SQL> heading "Désignation"

SQL>
```

1.2.1 Gestion du buffer

La dernière instruction SQL est stockée en mémoire (buffer).
Ce buffer peut être réutilisé dans plusieurs cas :

- **réexécution de la dernière commande SQL ou du dernier bloc PL/SQL.**
- **visualisation de la dernière commande.**
- **modification de la dernière commande.**
- **sauvegarde dans un fichier du buffer.**

La réexécution du buffer se fait par la commande RUN ou /.

LIST

Visualisation du buffer.

La ligne courante est indiquée par une astérisque (*). Il est possible de changer la ligne courante en ne listant qu'une seule ligne (on précise son numéro). La ligne listée devient alors la nouvelle ligne courante.

Syntaxe

`L[IST] [n|n m|n|n LAST|*|* n|* LAST|LAST]`

`n, m` **: numéro de ligne**

`*` **: ligne courante**

`LAST` **: dernière ligne**

Rq: Il est également possible de lister une seule ligne (qui devient alors la ligne courante) en tapant directement le numéro de la ligne.

Exemple

Saisie d'une commande SQL, visualisation du buffer, puis exécution :

```
SQL> l
  1  SELECT nocde, sum(qtecde*prix) as total
  2  from lignescde, produits
  3  where lignescde.refart=produits.refart
  4* group by nocde
SQL> l 2 3
  2  from lignescde, produits
  3* where lignescde.refart=produits.refart
SQL> l 2 LAST
  2  from lignescde, produits
  3  where lignescde.refart=produits.refart
  4* group by nocde
SQL> l 2
  2* from lignescde, produits
SQL> l * LAST
  2  from lignescde, produits
  3  where lignescde.refart=produits.refart
  4* group by nocde
SQL>
```

INPUT

Ajout de ligne au buffer après la ligne courante.

Syntaxe

I[NPUT] [texte]

Exemple

Ajout d'une ligne à la dernière commande :

```
SQL> select *
  2    from clients
  3   where nocli>15
  4
SQL> i
  4  order by nomcli
  5
SQL> l
  1  select *
  2    from clients
  3   where nocli>15
  4* order by nomcli
SQL>
```

APPEND

Ajout à la fin de la ligne courante.

Syntaxe

A[PPEND] texte

Exemple

Ajout de texte à la fin de la ligne 2 :

```
SQL> l
  1 select * from CLIENT
  2 where NOCLI>15
  3* order by NOM
SQL> 2
  2* where NOCLI>15
SQL> A and NOCLI<20
  2* where NOCLI>11 and NOCLI<20
SQL>
```

CHANGE

Modification de la ligne courante.

Syntaxe

C[HANGE]/ancien texte/[nouveau texte]

Si le nouveau texte est omis, il y a suppression de l'ancien texte.

Exemple

Modification de la première ligne :

```
SQL> l
  1 select * from clients
  2* order by NOCLI
SQL>1
  1* select * from clients
SQL>C/clients/COMMANDES
  1* select * from COMMANDES
SQL>
```

DEL

Suppression d'une ou de plusieurs lignes du buffer.

Syntaxe

DEL [n|n m|n|n LAST|*|* n|* LAST|LAST]

n, m : numéro de ligne

*** : ligne courante**

LAST : dernière ligne

Rq: Si la commande DEL est exécutée sans paramètres alors la ligne courante est supprimée.

Rq: Si la ligne courante est supprimée, alors c'est la ligne suivante (si elle existe) qui devient la nouvelle ligne courante.

Exemple

Suppression de la dernière ligne du buffer :

```
SQL> l

  1 select * from clients

  2 where NOCLI>10

  3* order by NOM

SQL> del

SQL> l

  1 select * from clients

  2* where NOCLI>10

SQL>
```

SAVE

Sauvegarde du buffer dans un fichier.

Syntaxe

```
S[AVE] nomfic [{CREATE/REPLACE/APPEND}]
```

Le fichier créé prend automatiquement l'extension .sql si aucune autre extension n'est précisée.

```
CREATE
```

(Option par défaut) crée un nouveau fichier.

```
REPLACE
```

Remplace le contenu d'un fichier existant.

```
APPEND
```

Ajoute le buffer à la fin d'un fichier existant.

GET

Chargement du buffer à partir d'un fichier.

Syntaxe

```
G[ET] nomfic [{LIST/NOLIST}]
```

```
LIST
```

(Option par défaut) liste le buffer.

NOLIST

Supprime l'affichage.

1.2.2 Utilisation de scripts

Il est possible de stocker des commandes SQL ou SQL*Plus dans des fichiers de texte soit par SAVE, soit par l'éditeur système, puis d'exécuter ces scripts par SQL*Plus. Ces fichiers de commandes sont reconnus avec l'extension par défaut .sql, ils peuvent contenir des commentaires.

EDIT

Appel de l'éditeur standard du système d'exploitation.

Syntaxe

```
[ED]IT [nomfic]
```

L'éditeur appelé est l'éditeur système sauf si la variable _EDITOR est valorisée (sous UNIX def _editor = vi, sous NT def _editor=notepad.exe).

Edit charge par défaut le fichier avec l'extension .sql. Si aucun nom de fichier n'est spécifié, Edit crée un fichier AFIEDT.BUF et recharge le buffer avec le contenu du fichier au retour dans SQL*Plus.

REM

Commentaires.

Syntaxe

```
REM[ARK] texte
```

Le texte sera ignoré à l'exécution.

Les commentaires peuvent aussi être indiqués par :

```
--
```

Le reste de la ligne situé après -- est ignoré par SQL*Plus.

```
/*... */
```

Le texte situé entre ces deux séparateurs est ignoré par SQL*Plus. Le commentaire peut comporter plusieurs lignes.

START, @, @@

Exécution des commandes contenues dans le fichier.

Syntaxe

```
STA[RT] nomfic ou @nomfic ou @@nomfic
```

Par défaut, l'extension .sql est prise pour le fichier.

Le nom du fichier doit contenir le chemin complet d'accès au fichier si ce dernier n'est pas dans le répertoire courant.

Par défaut, si seul le nom du fichier est indiqué derrière la commande START ou @ alors le fichier script est recherché dans le répertoire de travail de SQLPLUS. Le fait de lancer un script au travers de la commande @@ permet de garantir que le fichier sera recherché dans le même répertoire que celui du script qui demande l'exécution.

COPY

Il est possible de copier des données d'une table à une autre table située sur une base de données locale ou non. Pour cela on utilise la commande SQL*Plus COPY et les clauses FROM et TO pour préciser la source et la destination des données. Les données à copier sont issues d'une requête SELECT.

Syntaxe

```
COPY {FROM base| TO base| FROM base TO base}
{APPEND|CREATE|INSERT|REPLACE} table_de_destination
[(colonne, ...)] USING requête
```

FROM base

Permet de préciser le schéma qui contient les données à copier. Si la clause FROM est omise, alors l'origine des données est le schéma auquel SQL*Plus est connecté.

TO base

Permet de préciser le schéma de destination des données. Si la clause TO est omise, alors la destination des données est le schéma auquel SQL*Plus est connecté.

base

Sous ce terme sont regroupés le nom de l'utilisateur, son mot de passe et la chaîne de connexion pour accéder à une base de données distante.

APPEND

Les données issues de la requête sont insérées dans la table de destination. Si cette dernière n'existe pas, alors la table est créée avant l'insertion des données.

CREATE

Les données issues de la requête sont insérées dans la table de destination après la création de cette dernière. Si la table de destination était déjà présente avant l'exécution de la commande COPY, alors une erreur est retournée.

INSERT

Les données issues de la requête sont insérées dans la table de destination. Si la table de destination n'existe pas, alors COPY retourne une erreur.

REPLACE

Les données issues de la requête remplacent celles de la table de destination. Si la table de destination n'existe pas, alors elle est créée par la commande COPY.

Exemple

La table DEPT de l'utilisateur SCOTT va être copiée sous le schéma de l'utilisateur LIVRE. Les deux schémas se situent sur une base de données distante.

```
SQL> COPY FROM SCOTT/TIGER@DBHELIOS -
> TO LIVRE/ENI@DBHELIOS -
> APPEND dept -
> USING SELECT * FROM DEPT

Taille tableau extrait/lié est 15. (taille du tableau est 15)
Sera validé après opération. (COPYCOMMIT = 0)
Taille maximum (LONG) est 80. (longueur est 80)
La table DEPT est créée.

   4 lignes sélectionnées à partir de SCOTT@DBHELIOS.
   4 lignes insérées dans DEPT.
   4 lignes validées en DEPT à LIVRE@DBHELIOS.

SQL>
```

1.3 Gestion de l'environnement SQL*Plus

SQL*Plus permet de gérer directement (sans passer par le SQL) certaines fonctionnalités de la base, comme la visualisation des structures d'objet ou la copie de données d'une base à une autre. Il est également possible de personnaliser l'utilisation de SQL*Plus en modifiant des paramètres agissant sur le fonctionnement des commandes, sur l'affichage, ou sur le résultat des commandes.

DESCRIBE

Description de structure de table, vue, synonyme ou code stocké.

Syntaxe

```
DESC[RIBE] objet
```

SPOOL

Redirection des sorties.

Syntaxe

```
SPO[OL] {nomfic/OFF/OUT}
```

`nomfic`

Nom du fichier recevant les sorties (extension .lst par defaut).

`OFF`

Fermeture du fichier.

`OUT`

Fermeture du fichier et sortie sur l'imprimante.

SHOW

Visualisation de l'environnement.

Syntaxe

```
SHO[W] {ALL/paramètre}
```

SET

Paramètres d'environnement.

Syntaxe

`SET parametre valeur`

La valeur par défaut des paramètres est la première des valeurs possibles indiquées :

AUTOCOMMIT {OFF/ON/IMMEDIATE/n}

Validation automatique.

CMDSEP {;/OFF/ON/c}

Séparateur de commandes multilignes.

FEEDBACK {6/n/OFF/ON}

Affichage du nombre d'enregistrements selectionnés à partir de n.

LONG {80/n}

Format maximum des colonnes de type LONG.

NULL {texte}

Représentation des valeurs NULL.

PAUSE {OFF/ON/texte}

Activation de la pause en fin de page.

SPACE {1/n}

Nombre de caractères séparant les colonnes à l'affichage.

SQLCASE {MIXED/LOWER/UPPER}

Conversion en majuscules ou minuscules des commandes avant exécution.

SQLCONTINUE {>/texte}

Prompt de la énième ligne.

SQLNUMBER {ON/OFF}

Numérotation du prompt de énième ligne.

SQLPROMPT {SQL/texte}

Prompt de la première ligne.

SQLTERMINATOR {;/c/OFF/ON}

Caractère de fin de commande.

ECHO {OFF/ON}

Affichage des commandes d'un script à l'exécution.

TERMOUT {ON/OFF}

Affichage du résultat des commandes d'un script.

HELP

Accès au système d'aide (s'il a été installé).

Syntaxe

```
HELP [commande]
```

HOST

Accès au système d'exploitation.

Syntaxe

```
HOST [commande S.E.]
```

Rq: Avec certains systèmes d'exploitation, il est possible d'utiliser un caractère générique pour lancer une commande système depuis SQL*Plus. Comme ce caractère dépend du système, il faut consulter la documentation Oracle relative à la plate-forme d'installation pour connaître ce caractère.

2 Présentation des données

SQL*Plus offre quelques fonctionnalités pour gérer des variables, saisir des valeurs et mettre en page des résultats de requêtes.
Cela permet d'utiliser de manière plus conviviale le SQL ou le PL/SQL et de créer de véritables procédures de manière simple.

2.1 Gestion des variables

Il est possible de gérer deux sortes de variables :

- **les variables utilisateur, utilisées par SQL*Plus ou pour de la substitution de texte dans des commandes SQL et SQL*Plus avant leur exécution.**
- **les variables de lien utilisées en interface avec le PL/SQL.**

2.1.1 Les variables de substitution

DEFINE

Création d'une variable utilisateur.

Syntaxe

```
DEF[INE] [variable = texte]
```

Sans paramètre, DEFINE permet de visualiser les variables existantes.

Les variables créées sont obligatoirement de type caractère.

Certaines variables ont des noms et des utilisations réservées :

_EDITOR

Éditeur appelé par EDIT.

_O_VERSION

Version ORACLE.

_O_RELEASE

Numéro de "release" ORACLE.

_CONNECT_IDENTIFIER

Identifiant utilisé pour la connexion.

_DATE

Date courante.

On peut définir 1024 variables au maximum.

UNDEFINE

Suppression d'une variable utilisateur.

Syntaxe

```
UNDEF[INE] variable
```

Exemple

Création des variables x et _editor, visualisation des variables, suppression de x.

```
SQL> define x = abcdef

SQL> def _EDITOR = vi

SQL> DEF

DEFINE _EDITOR          ="vi"(CHAR)

DEFINE X               ="abcdef" (CHAR)

SQL> undef x
```

ACCEPT

Saisie d'une variable utilisateur.

Si la variable n'a pas été définie, elle est créée.

Syntaxe

```
ACCEPT variable [NUM/CHAR] [PROMPT texte] [HIDE]
```

```
NUM/CHAR
```

Caractères autorisés pour la valeur de la variable (chiffres uniquement ou alphanumérique). CHAR par défaut.

```
PROMPT texte
```

Affichage du texte avant la saisie.

```
HIDE
```

Frappe aveugle.

Exemple

Saisie des variables V_code et V_nom :

```
SQL> accept V_NOM
(saisie par l'utilisateur de : Dupont)
SQL> accept V_CODE NUM prompt "Code ?"
Code ? (saisie de : 15)
SQL> def
DEFINE V_NOM      = "Dupont" (CHAR)
DEFINE V_CODE     = 15 (NUMBER)
SQL>
```

&nom

Utilisation d'une variable de substitution.

Syntaxe

```
&variable
```

Si la variable n'existe pas, une saisie est demandée.

Exemple

Saisie de variables et utilisation dans une commande SQL :

```
SQL> accept NUMERO NUMBER prompt "Numero client ?"
Numero client ? 15
SQL> accept NOM prompt "Nom client ?"
Nom client ? TOTO
SQL> def tb = clients
SQL> select * from &tb where
  2> Nocli > &NUMERO and
  3> NOM = '&NOM';
ancien  1 : select * from &tb where
nouveau 1 : select * from client where
ancien  2 : nocli  > &NUMERO
nouveau 2 : nocli  >15
ancien  3 : NOM    ='&NOM' ;
nouveau 3 : NOM    ='TOTO' ;
aucune ligne sélectionnee
SQL>
```

&&nom

Permet en plus de créer la variable.

Syntaxe

```
  &&variable
```

Exemple

Lorsque la requête est réexécutée, la valeur de la variable n'est pas redemandée.

```
SQL> select * from &&NomTab
  2  /
Entrez une valeur pour nomtab: LIGCDES
ancien   1: select * from &&NomTab
nouveau  1: select * from LIGCDES


aucune ligne sélectionnée


SQL> run
  1* select * from &&NomTab
ancien   1: select * from &&NomTab
nouveau  1: select * from LIGCDES


aucune ligne sélectionnée


SQL>
```

2.1.2 Les variables hôtes

VARIABLE

Création d'une variable pour utilisation dans un bloc PL/SQL.

Syntaxe

```
VAR[IABLE] [nom [NUMBER/CHAR(n)]]
```

Sans paramètre, VARIABLE affiche les caractéristiques des variables existantes.

Une variable PL/SQL ne peut être valorisée que dans un bloc PL/SQL par des instructions spécifiques au PL/SQL, elle est considérée par le PL/SQL comme une variable de l'hôte (préfixée de :).

PRINT

Affichage du contenu de la variable PL/SQL.

Syntaxe

```
PRINT variable
```

Exemple

```
SQL> var ncli number
SQL> variable
```

```
variable   ncli

type de données   NUMBER

SQL> begin

  2    select count(*) into :ncli

  3      from clients;

  4  end;

  5  /


Procédure PL/SQL terminée avec succès.


SQL> print ncli


      NCLI

----------

         7


SQL>
```

2.2 Présentation des résultats

Le résultat des requêtes peut être mis en forme de manière simple en utilisant des commandes SQL*Plus.
Ces commandes sont généralement des déclarations qui doivent être faites avant les instructions SELECT et qui doivent être annulées ou désactivées après la requête.

2.2.1 Contrôle du déroulement des scripts

PAUSE

Définition d'une suspension d'exécution.

Syntaxe

```
PAUSE [texte]
```

PROMPT

Affichage de texte.

Syntaxe

```
PROMPT [texte]
```

2.2.2 En-tête et pied de page

Déclaration ou annulation d'un en-tête : TTITLE.
Déclaration ou annulation d'un pied de page : BTITLE

BTITLE, TTITLE

Syntaxe

{BTITLE/TTITLE} [option [texte/variable]]/ [OFF/ON]

Les différentes options sont :

COL n

Place le texte en colonne n.

SKIP n

Saute n lignes.

TAB n

Saute n colonnes.

LEFT/CENTER/RIGHT

Alignement.

BOLD

Caractère gras.

FORMAT

Format d'affichage composé de :

9 remplace un chiffre (ex. : format '9999' pour 4 chiffres).

0 affiche les zéros non significatifs (ex : '00999').

$ est le préfixe du signe $.

B remplace les zéros non significatifs par des espaces.

MI/PR : MI affiche - après une valeur négative, PR affiche les valeurs négatives entre crochets.

, insère une virgule.

. positionne la marque décimale (ex. : '999.99' pour 3 entiers 2 décimales).

V multiplie par 10 fois le nombre de décimales.

EEEE est une notation scientifique.

DATE est le format date.

An permet un affichage sur n caractères.

Les variables prédéfinies :

SQL.PNO

Page courante.

SQL.LNO

Ligne courante.

SQL.RELEASE

Numéro de release ORACLE.

SQL.SQLCODE

Dernier code erreur.

SQL.USER

Utilisateur courant.

2.2.3 Rupture

BREAK

Déclaration d'action à effectuer lors d'un changement de valeur de la colonne.

Syntaxe

```
BREAK [ON{colonne/expression/ROW/REPORT}
[{SKIP{n/PAGE}/DUPLICATES}]...]
```

2.2.4 Format de colonne

COLUMN

Déclaration des attributs des colonnes sélectionnées.

Syntaxe

```
COLUMN [{colonne/expression} option]
```

Les différentes options sont :

ALIAS nom

Nom alternatif.

LIKE alias

Copie les attributs de l'alias.

CLEAR

Réinitialise les attributs.

FOLD_AFTER

Insère un retour chariot après la colonne.

FOLD_BEFORE

Insère un retour chariot avant la colonne.

FORMAT format

Format de la valeur affichée.

HEADING texte

En-tête de colonne.

JUSTIFY{LEFT/CENTER/RIGHT}

Justification.

NEWLINE

Idem à FOLD_BEFORE.

NEW_VALUE variable

Place la valeur de la colonne dans la variable pour l'afficher dans l'en-tête de page.

OLD_VALUE variable

Idem à NEW_VALUE pour le pied de page.

NULL c

Texte pour les valeurs nulles.

NOPRINT/PRINT

Affichage ou non de la colonne.

WRAPPED/WORD_WRAPPED/TRUNCATED

Césure des lignes trop longues.

OFF

Désactivation de la définition.

ON

Réactivation de la définition.

2.2.5 Calcul statistique

COMPUTE

Déclaration d'un calcul à effectuer au changement de valeur de l'expression (qui doit avoir été déclarée par un BREAK).

Syntaxe

```
COMPUTE fonction OF {expression/colonne}
ON {expression/colonne/REPORT/ROW}
```

Les fonctions qu'il est possible d'utiliser sont :

AVG

Moyenne des valeurs non nulles.

COUNT

Dénombrement des valeurs non nulles.

MAXIMUM

Valeur maxi.

MINIMUM

Valeur mini.

NUMBER

Nombre de lignes.

STD

Écart-type.

SUM

Somme des valeurs.

VARIANCE

Variance.

2.2.6 Annulation des déclarations

CLEAR

Syntaxe

```
CLEAR {BREAKS/COLUMNS/COMPUTES/TIMING/BUFFER/SQL/SCREEN}
```

2.3 Environnement et traitement des erreurs

Un certain nombre de paramètres d'environnement permettent de gérer le fonctionnement des commandes de mise en page. Ces paramètres sont généralement valorisés en début de script et annulés en fin de script.

2.3.1 Statistiques de temps

TIMING

Syntaxe

```
TIMING [{START texte/SHOW/STOP}]
```

2.3.2 Traitement des erreurs

En cas d'erreurs soit du système d'exploitation, soit du SQL, SQL*Plus les ignore, ce qui peut être gênant dans des enchaînements de commandes ou pour l'intégrité des données.

La déclaration WHENEVER permet de prévenir ces erreurs.

WHENEVER

Syntaxe

```
WHENEVER {OSERROR/SQLERROR} {EXIT[SUCCESS/
FAILURE/OSCODE/"][COMMIT/ROLLBACK]/CONTINUE
[COMMIT/ROLLBACK/NONE]}
```

EXIT

Sort de SQL*Plus en envoyant le code erreur spécifié.

CONTINUE

Ne sort pas de SQL*Plus.

COMMIT ou ROLLBACK

Force une validation ou une annulation avant de sortir ou de continuer.

NONE

Ne fait rien avant de sortir ou de continuer.

SET ERRORLOGGING

Syntaxe simplifiée

```
SET ERRORL[OGGING] {ON | OFF} [TRUNCATE] [IDENTIFIER identifier]
```

Active ou désactive l'enregistrement dans une table des erreurs qui surviennent dans SQL*Plus.

Par défaut, les erreurs sont enregistrées dans une table nommée SPERRORLOG ; si cette table n'existe pas, elle est créée automatiquement.

TRUNCATE

Vide la table lors de l'activation de l'enregistrement.

IDENTIFIER

Chaîne de caractères utilisée pour identifier les erreurs.

2.3.3 Paramètres d'environnement

SET

Paramètres liés à l'affichage.

Syntaxe

```
SET parametre valeur
```

DEFINE {'&'/c/ON/OFF}

caractère utilisé pour les substitutions de variables.

EMBEDDED {OFF/ON}

chaque rapport débute sur une nouvelle page.

HEADSEP {|/c/ON/OFF}

caractère définissant un en-tête de colonne multiligne.

LINESIZE {80/n}

nombre de caractères par ligne.

NEWPAGE {1/n/NONE}

nombre de lignes entre le début de page et l'en-tête défini.

NUMFORMAT {format}

format par défaut des numériques.

NUMWIDTH {10/n}

taille par défaut des numériques.

PAGESIZE {14/n}

nombre de lignes entre l'en-tête et le bas de page.

SCAN {ON/OFF}

contrôle la présence des variables de substitution.

SHOWMODE {ON/OFF}

affichage des valeurs des variables en cas de changement.

TAB {OFF/ON}

utilisation des tabulations.

TIME {OFF/ON}

affichage de l'heure avant chaque commande.

TIMING {OFF/ON}

affichage des statistiques de temps.

TRIMOUT {ON/OFF}

suppression des espaces en fin de ligne.

UNDERLINE {-/c/ON/OFF}

caractère de soulignement.

VERIFY {ON/OFF}

liste la commande avant son exécution (en cas de substitution de variable).

WRAP {ON/OFF}

césure des lignes.

Rq: La commande SET possède de nombreuses autres options mais seules les plus courantes sont documentées dans cet ouvrage.

Exemple

Script de mise en page, extraction et sauvegarde dans un fichier (confcde.lis) d'une confirmation de commande dont le numéro est saisi par l'utilisateur.

```
rem Edition de la confirmation d'une commande
rem Environnement
set echo off
set feed off
set pagesize 60
set newpage 0
set linesize 80

rem Saisie du numero de commande

accept VCDE number prompt 'No de commande à éditer ? '

rem Mise en page

set term off

column REFART heading "Reference|article"
column DESIGNATION heading "Désignation"
column PRIXUNITHT heading "Prix H.T."
column QTECDE heading "Qté cde."
column NOCDE noprint new_value VNOCDE
column NOMCLIENT noprint new_value VNOM

break on NOCDE skip page

compute sum of Mt on NOCDE

ttitle center "Confirmation de commande" -
      skip 2 left "Numero : " VNOCDE -
      skip 1 left "Client : " VNOM skip 3

btitle tab 40 "Page " sql.pno

spool confcde.lis

rem Extraction

select COMMANDES.NOCDE,NOMCLIENT,LIGNESCDE.REFART,DESIGNATION,
      PRIXUNITHT,QTECDE,PRIXUNITHT*QTECDE "Mt"
      from COMMANDES,LIGNESCDE,CLIENTS,ARTICLES
      where COMMANDES.NOCDE = &VCDE
      and  COMMANDES.NOCLIENT = CLIENTS.NOCLIENT
      and  COMMANDES.NOCDE = LIGNESCDE.NOCDE
    and  LIGNESCDE.REFART = ARTICLES.REFART
/
spool out

rem Annulation de la mise en page
```

```
ttitle off
btitle off
clear breaks
clear columns
clear compute
set pagesize 14
set newpage 1
set feed on
set echo on
set term on
```

Résultat du script précédent si N° de commande saisi est 1210 :

```
                    Confirmation de commande


Numero :    1210
Client : DUPONT S.A.


Reference
article  Désignation        Prix H.T.   Qté cde.   Mt
---------- ---------------- ---------- --------- -------
AB10       Tapis de Chine     1500        3       4500
CD50       Chaîne HIFI          735,4     4       2941,6
                                                 -------
                                                   7441,6
                                Page 1
```

2.4 Création d'un rapport au format HTML

SQL*Plus offre la possibilité de créer un rapport au format HTML en utilisant les commandes SET MARKUP et SPOOL. Les informations du rapport sont insérées à l'intérieur de la balise HTML <PRE>, et donc l'affichage au sein de l'explorateur Internet est exactement le même que celui perçu dans SQL*Plus.

Syntaxe

```
SET MARK[UP] HTML [ON|OFF] [HEAD 'texte'] [BODY 'texte'][TABLE 'texte']
[ENTMAP {ON|OFF}] [SPOOL {ON|OFF}] [PRE[FORMAT]{ON|OFF}]
```

Les choix par défaut des options sont soulignés.
La commande SET MARKUP permet à SQL*Plus de générer un résultat au format HTML et non plus au simple format texte. L'intérêt de fournir directement les données au format HTML n'est plus à démontrer car ce format permet une présentation simple à mettre en place et soignée tout en restant indépendante de l'outil utilisé pour visualiser le résultat.
Les différents paramètres de cette commande sont :

HTML

(OFF par défaut) Positionner à ON la valeur de cette option permet d'activer la génération des résultats au format HTML.

HEAD

Permet de personnaliser l'en-tête de la page HTML comme par exemple le fait de définir un titre à l'aide des balises <TITLE> et </TITLE>. C'est également à cet endroit que l'on pourra définir les styles à appliquer aux différentes balises.

BODY

L'option BODY permet de définir les attributs de la balise BODY. Par défaut aucun attribut n'est sélectionné mais il est toujours possible de le faire par cette option.

TABLE

Permet de spécifier les attributs du tableau. Par défaut, le tableau est défini comme possédant une largeur de 90% (WIDTH='90%') et la largeur de la bordure est fixée à 1 (border='1').

ENTMAP

(ON par défaut) Positionner à OFF la valeur de cette option permet d'interpréter les caractères HTML correctement, ce qui peut être le cas pour <, >," et &. Avec cette précaution, ces caractères ne sont pas remplacés par leur entité respective &lt, &gt,&quot et &amp.

SPOOL

(OFF par défaut) Positionner à ON la valeur de cette option permet d'ajouter les balises <HTML> et <BODY> ainsi que les balises fermant correspondant en début et fin de fichier. Ce paramètre n'est en aucun cas équivalent à la commande SPOOL qui permet de rediriger les informations affichées à l'écran vers un fichier.

PREFORMAT

(OFF par défaut) indique à SQL*Plus que les résultats sont présentés sous forme de tableau. Positionner à ON la valeur de cette option permet de préciser les informations indiquées après une balise <PRE>.

Rq: Afin de présenter au mieux le script SQL*Plus qui contient la commande SET MARKUP, il est fortement recommandé de saisir cette commande sur plusieurs lignes en prenant soin de ne pas oublier le caractère de continuation (-) des instructions SQL*Plus.

Le script ci-dessous illustre comment il est possible de produire un résultat au format HTML depuis SQL*Plus.

```
rem Exemple de génération de rapport en HTML
set echo on
set markup html on spool on entmap on -
  head "<TITLE>Les commandes</TITLE> -
  <STYLE type='text/css'> -
  <!-- BODY{background:#FFFFC6} --> -
  </style>"-
  BODY "TEXT='#FF00FF'" -
  TABLE "BORDER='2'" -
  preformat off
column nocde heading 'N°'
column nocli heading 'Client'
column datecde heading 'Date'
column etatcde heading 'Etat' format A4
spool test.html
select nocde, nocli, datecde, etatcde from commandes;
spool off
set markup html off spool off
```

L'exécution de ce script donne le résultat suivant :

```
SQL> select nocde, nocli, datecde, etatcde from commandes;

N°  Client  Date  Etat
100  15  23/07/02  EC
1301  15  22/07/02  EC
1210  15  20/07/02  SO
1250  35  19/07/02  EC
```

```
1230  35  18/07/02  SO

SQL> spool off
```