Utilitaire SQL*PLUS

Table des matières

1 Mise en forme des SELECT

SQL*PLUS possède de nombreuses commandes pour mettre en forme les sorties écran des SELECT. En particulier, des commandes puissantes permettent d'obtenir des totaux et des sous-totaux sur les colonnes numériques. Ces commandes sortent du cadre de ce cours. Nous verrons seulement la commande "COLUMN" qui est bien utile pour donner la largeur des colonnes des SELECT pour qu'elles ne soient pas trop larges.

Largeur de 12 caractères pour la colonne NOME :

```
COLUMN NOME FORMAT A12
```

2 Commandes utilitaires de SQL*PLUS

2.1 Éditeur de la machine hôte

La commande ED appelle un éditeur de texte externe à Oracle en lui passant le contenu du buffer SQL.

Il est possible de spécifier un autre éditeur que l'éditeur standard du système d'exploitation ("ed" pour Unix) ; pour travailler avec l'éditeur emacs, on peut taper :

```
DEF _editor = emacsclient
```

Si on travaille avec emacs, il faut utiliser le mode "client-serveur" d'*emacs* sinon on lance un nouvel *emacs* à chaque appel de ED.

2.2 Commande RUN, commande /

La commande RUN (abréviation : R) liste la commande contenue dans le buffer SQL et en lance l'exécution.

La commande / lance l'exécution de la commande dans le buffer SQL sans la lister.

2.3 Commande HOST

`HOST` *commande*

(abréviation "`!` *commande*") permet d'exécuter la *commande* du système d'exploitation tout en restant à l'intérieur de SQLPLUS.

Exemple 2.1

`HOST ls -l` (ou `!ls -l`)

va faire s'afficher sur votre écran les fichiers existants dans votre répertoire courant.

On peut taper "HOST" sans rien préciser à la suite pour appeler un shell que l'on peut quitter en tapant "exit" pour se retrouver sous SQLPLUS.

2.4 Sauvegarde de la session de travail

`SPOOL` *fichier*

enregistre dans le *fichier* (*fichier*.lst si *fichier* n'a pas d'extension ".") les commandes tapées ensuite par l'utilisateur, et les réponses d'Oracle.

`SPOOL OFF`

arrête cette sauvegarde.

3 Aide

`HELP` *commande-SQL*
est une commande SQL*PLUS qui affiche de l'aide sur *commande-SQL*.

Pour faire afficher de l'aide sur un des modules fournis par Oracle, on peut aussi lancer la commande *oradocm* placée dans le répertoire d'Oracle sous le répertoire `orainst`. Une grande partie de la documentation d'Oracle est disponible sous forme de documents hypertextes (double clic rapide pour suivre un "lien").

3.1 Fichiers de commandes SQL

Il est possible de constituer des fichiers contenant des ordres SQL.

3.1.1 Sauvegarde du contenu du buffer dans un fichier

`SAVE` *fichier* `[APPEND]`
sauvegarde le buffer dans *fichier*, en le créant s'il n'existe pas encore, ou en écrasant son contenu dans le cas contraire. Le contenu du buffer est ajouté en fin de fichier si l'option APPEND est précisée.

L'extension "`.SQL`" est ajoutée par défaut au nom du fichier. Un ordre / est automatiquement ajouté à la fin de chaque ordre (pour être exécutés automatiquement par la commande START étudiée en 3.4).

3.2 Lecture dans le buffer d'un fichier de commandes

`GET` *fichier*
lit le contenu de *fichier*, et l'injecte dans le buffer SQL que l'on peut ensuite modifier ou/et exécuter. S'il y a un / final dans *fichier*, il est exécuté.

3.3 Fichier login.sql

S'il existe un fichier nommé "`login.sql`" dans le répertoire courant lorsque l'utilisateur tape la commande "*sqlplus*", ce fichier (qui doit contenir des commandes SQL ou SQLPLUS) est exécuté automatiquement à l'entrée dans SQLPLUS.

Exemple de contenu de fichier `login.sql` :

```
set pagesize 22
set pause on
set pause "Taper <Return> pour la suite"
def _editor = emacs
```

3.4 Commandes START et @

`START` *fichier* `[`*arguments...*`]`
lance l'exécution des commandes de *fichier*. Tout se passe comme si les commandes du fichier étaient tapées au clavier. Les arguments remplacent les "`&1, &2,...`" contenus dans le texte de *fichier*.

Les ordres SQL contenus dans *fichier* ne sont pas exécutés automatiquement. Il faut que *fichier* contienne l'ordre explicite d'exécuter la commande SQL, soit en la terminant par un point-virgule, soit en la faisant suivre d'un / ou d'un RUN sur la ligne suivante.

La commande "`@`" est semblable à START.

4 Éditeur SQL*PLUS

Cet éditeur est très rudimentaire et on ne l'utilise en général que pour des petites modifications très simples. Sinon, il vaut mieux utiliser la commande "ed" de SQL*PLUS étudiée en 2.1.

4.1 Buffer de commande

SQLPLUS traite différemment les ordres qui lui sont propres et ceux qui appartiennent au langage SQL :

- les commandes SQL sont mémorisées dans un tampon. Elles ne sont exécutées que si la commande est terminée par un point virgule,
- les commandes non SQL ajoutées par SQLPLUS sont exécutées dès que l'on a frappé la touche RETURN.

Pour laisser la commande SQL dans le buffer et revenir au prompt `SQL>` sans exécuter la commande, il suffit de laisser la dernière ligne blanche. En effet, tant qu'on n'a pas terminé sa commande par un point virgule ou une ligne vierge, SQLPLUS demande des lignes supplémentaires (et le prompt est alors remplacé par le numéro de ligne). Par exemple,

```
SQL> SELECT *
  2  FROM DEPT
  3  ;
```

Cette mise en buffer des commandes SQL permet de rappeler la dernière commande pour l'exécuter à nouveau, l'éditer ou la sauvegarder dans un fichier (voir commande RUN en 2.2).

```
SQL> SELECT *
  2  FROM DEPT
```

```
  3
SQL> RUN
  1  SELECT *
  2* FROM DEPT

DEPT    NOMD            LIEU
------- --------------- --------------
10      FINANCES        PARIS
20      RECHERCHES      GRENOBLE
30      VENTES          LYON
40      FABRICATION     ROUEN
```

Il n'existe qu'un seul buffer de commande SQL. Chaque nouvel ordre SQL efface le précédent.

Les ordres SQLPLUS n'appartenant pas à SQL n'effacent pas le buffer puisqu'ils n'y sont pas mémorisés.

4.1.1 Commande LIST

La commande LIST (abréviation : L) permet d'afficher le contenu du buffer. Chaque ligne est précédée de son numéro. La dernière est précédée de "*" : c'est la *ligne courante*. La ligne courante est la ligne sur laquelle les autres commandes de l'éditeur SQLPLUS agiront. La commande LIST ramène toujours la ligne courante à la dernière ligne du buffer.

Exemple :

```
SQL> LIST
  1  SELECT NOMD
  2* FROM DEPT
SQL>
```

Il est possible de faire suivre d'un nombre le mot clé LIST. SQLPLUS affiche alors uniquement la ligne du buffer qui correspond à ce numéro, c'est alors elle qui devient le ligne courante.

Exemple :

```
SQL> L2
  2* FROM DEPT
```

4.1.2 Commande CHANGE

La commande CHANGE (abréviation : C) permet de remplacer une chaîne de caractères par une autre dans la ligne courante.

`C/chaîne1/chaîne2/`

remplace la 1ère occurrence de *chaîne1* par *chaîne2* dans la ligne courante.

Exemple :

```
SQL> L1
  1* SELECT *
SQL> C/*/LIEU/
  1* SELECT LIEU
```

Remarque : le séparateur "/" peut être remplacé par n'importe quel caractère ne figurant ni dans chaîne1 ni dans chaîne2 (si c'est une lettre, il faut un espace entre le "CHANGË et le premier séparateur).

Exemple :

```
SQL> L1
  1* SELECT LIEU
SQL> C,LIEU,*,
  1* SELECT *
```

Si la deuxième chaîne est vide ou absente, la première chaîne est supprimée :

```
SQL> L*
  1* SELECT *
SQL> C:E::
  1* SLECT *
```

Le dernier séparateur n'est pas nécessaire, sauf si la chaîne de caractères se termine par des blancs.

Exemple :

```
SQL> L*
  1* SLECT *
SQL> C:SL:SEL
  1* SELECT *
```

4.2 Commande APPEND

La commande APPEND (abréviation : A) permet d'ajouter du texte en fin de la ligne courante.

`A texte`

Exemple :

```
SQL> L*
  1* SELECT LIEU
SQL> A , DEPT
  1* SELECT LIEU, DEPT
```

L'espace entre la commande et le texte est facultatif si le texte ne commence pas par une lettre. Il n'est pas considéré comme faisant partie du texte.

Exemple :

```
SQL> L*
  1* SELECT LIEU, DEPT
SQL> A,
  1* SELECT LIEU, DEPT,
```

Par contre, un deuxième espace sera considéré comme faisant partie du texte et sera inséré dans la ligne courante.

Exemple :

```
SQL> L*
  1* SELECT LIEU, DEPT,
SQL> A   NOMD
  1* SELECT LIEU, DEPT, NOMD
```

4.3 Commande INPUT

La commande INPUT (abréviation : I) permet d'insérer une ou plusieurs lignes après la ligne courante.

Le numéro de la première ligne insérée s'affiche. Il est alors possible de saisir plusieurs lignes. L'insertion se termine par la saisie d'une ligne vide.

Exemple :

```
SQL> L*
  1* SELECT DEPT,
SQL> I
  2i NOMD,
  3i LIEU
  4i
SQL> L
  1 SELECT DEPT,
  2 NOMD,
  3 LIEU
  4* FROM DEPT
```

On peut saisir le texte de la ligne à insérer directement derrière la commande I (après un espace si le texte commence par une lettre). L'insertion se termine à la fin de la ligne.

Exemple :

```
SQL> L3
  3* LIEU
SQL> I EMPLACEMENT
SQL> L
  1 SELECT DEPT,
  2 NOMD,
  3 LIEU
  4 EMPLACEMENT
  5* FROM DEPT
```

4.4 Commande DEL

La commande DEL supprime la ligne courante.

La nouvelle ligne courante est la ligne suivant la ligne supprimée (ou la ligne précédente si la ligne supprimée était la dernière).

Exemple :

```
SQL> L2
  2* NOMD,
SQL> DEL
SQL> DEL
SQL> DEL
SQL> L
  1  SELECT DEPT,
  2* FROM DEPT
```