
Rapports et fonctions de base

Bases de données

Patricia Serrano Alvarado

Rapports sur sqlplus

- Le résultats des requêtes posées sur sqlplus sont montrées dans un format général

```
SQL> select *
      2  from inscriptions;

NOM      COMP      MONTANT
-----
Bob      WRC        200
Bob      IRM         50
Gaston  WRC        100
Gaston  RIR        600
Tom      WRC       1000
Tom      IRM       1000
Tom      RIR       2000
Zoe      IRM       1000
Pok      WRC         500
Pok      RIR       1500
Gaston  IRM         200

NOM      COMP      MONTANT
-----
Pikpik  WRC
Pyroli  WRC
Zoe     WRC         250
Toto    WRC       1500
Bob     NAT         520
Tom     NAN         520

17 rows selected.
```

```
SQL> select *
      2  from competitions;

COMP  VILLE      PAYS      DATC
-----
WRC   Boston     USA       03-MAR-00
IRM   Paris      France    20-JUN-00
RIR   Tokyo      Japan     15-MAR-00
NAN   Nantes     France    15-JUN-05
NAT   Nantes     France    15-JUN-05

SQL>
```

Rapports sur sqlplus

- Sqlplus fournit de fonctions pour la réalisation de rapports de type texte

Command	Definition
remark	Tells SQL*Plus that the words to follow are to be treated as comments, not instructions.
set headsep	The heading separator identifies the single character that tells SQL*Plus to split a title into two or more lines.
ttitle	Sets the top title for each page of a report.
btitle	Sets the bottom title for each page of a report.
column	Gives SQL*Plus a variety of instructions on the heading, format, and treatment of a column.
break on	Tells SQL*Plus where to put spaces between sections of a report, or where to break for subtotals and totals.
compute sum	Makes SQL*Plus calculate subtotals.
set linesize	Sets the maximum number of characters allowed on any line of the report.
set pagesize	Sets the maximum number of lines per page.

Rapports sur sqlplus

set newpage	Sets the number of blank lines between pages.
spool	Moves a report you would normally see displayed on the screen into a file, so you can print it.
/* */	Marks the beginning and ending of a comment within a SQL entry. Similar to remark .
--	Marks the beginning of an inline comment within a SQL entry. Treats everything from the mark to the end of the line as a comment. Similar to remark .
set pause	Makes the screen display stop between pages of display.
save	Saves the SQL query you're creating into the file of your choice.
host	Sends any command to the host operating system.
start or @	Tells SQL*Plus to follow (execute) the instructions you've saved in a file.
edit	Pops you out of SQL*Plus and into an editor of your choice.
define _editor	Tells SQL*Plus the name of the editor of your choice.
exit or quit	Terminates SQL*Plus.

Processus de création de rapports sur sqlplus

Création d'un fichier
mon_fichier.sql
avec mon éditeur

```
Set linesize 80
Set pagesize 60
Spool rapport.lst
Select *
From competitions;
Spool off
```

Exécution du fichier
mon_fichier.sql
dans sqlplus

```
SQL>@mon_fichier

COMP VILLE      PAYS      DATC
-----
WRC  Boston     USA       03-MAR-00
IRM  Paris      France    20-JUN-00
RIR  Tokyo      Japan     15-MAR-00
NAN  Nantes     France    15-JUN-05
NAT  Nantes     France    15-JUN-05
```

Rapport créé
avec comme nom rapport.lst
visualisable et imprimable
avec votre éditeur de texte

```
COMP VILLE      PAYS      DATC
-----
WRC  Boston     USA       03-MAR-00
IRM  Paris      France    20-JUN-00
RIR  Tokyo      Japan     15-MAR-00
NAN  Nantes     France    15-JUN-05
NAT  Nantes     France    15-JUN-05
```

Exemple de rapport

- Fait avec le fichier `activite.sql`

Thu Apr 04

Checkout Log for 1/1/02-3/31/02

page 1

NAME	TITLE	CHECKOUTD	RETURNEDD	Days Out

DORAH TALBOT	EITHER/OR	02-JAN-02	10-JAN-02	8.00
	POLAR EXPRESS	01-FEB-02	15-FEB-02	14.00
	GOOD DOG, CARL	01-FEB-02	15-FEB-02	14.00
	MY LEDGER	15-FEB-02	03-MAR-02	16.00

avg				13.00
EMILY TALBOT	ANNE OF GREEN GABLES	02-JAN-02	20-JAN-02	18.00
	MIDNIGHT MAGIC	20-JAN-02	03-FEB-02	14.00
	HARRY POTTER AND THE GOBLET OF FIRE	03-FEB-02	14-FEB-02	11.00

avg				14.33
FRED FULLER	JOHN ADAMS	01-FEB-02	01-MAR-02	28.00
	TRUMAN	01-MAR-02	20-MAR-02	19.00

avg				23.50
GERHARDT KENTGEN	WONDERFUL LIFE	02-JAN-02	02-FEB-02	31.00
	MIDNIGHT MAGIC	05-FEB-02	10-FEB-02	5.00
	THE MISMEASURE OF MAN	13-FEB-02	05-MAR-02	20.00

avg				18.67
JED HOPKINS	INNUMERACY	01-JAN-02	22-JAN-02	21.00
	TO KILL A MOCKINGBIRD	15-FEB-02	01-MAR-02	14.00

avg				17.50
PAT LAVAY	THE SHIPPING NEWS	02-JAN-02	12-JAN-02	10.00
	THE MISMEASURE OF MAN	12-JAN-02	12-FEB-02	31.00

avg				20.50
ROLAND BRANDT	THE SHIPPING NEWS	12-JAN-02	12-MAR-02	59.00
	THE DISCOVERERS	12-JAN-02	01-MAR-02	48.00
	WEST WITH THE NIGHT	12-JAN-02	01-MAR-02	48.00

avg				51.67

avg				22.58

from the Bookshelf

Le fichier activite.sql

```
rem Bookshelf activity report ← 1

set headsep ! ← 2

ttitle 'Checkout Log for 1/1/02-3/31/02' ← 3
btitle 'from the Bookshelf'

column Name format a20 ← 4
column Title format a20 word_wrapped ← 5
column DaysOut format 999.99 ← 6
column DaysOut heading 'Days!Out' ← 7

break on Name skip 1 on report ← 8
compute avg of DaysOut on Name ← 9
compute avg of DaysOut on report

set linesize 80 ← 10
set pagesize 60
set newpage 0
set feedback off

spool activity.lst ← 11

select Name, Title, CheckoutDate, ReturnedDate,
       ReturnedDate-CheckoutDate as DaysOut /*Count Days*/ ← 12
  from BOOKSHELF_CHECKOUT
 order by Name, CheckoutDate;

spool off
```

Description de `activite.sql`

1. `remark | rem | -- | /* */`
2. Set `headsep`
 - De `heading separator` permet de définir un caractère servant à découper un titre ou entête de colonne
3. `ttitle` et `btitle`
 - De `top title` permet de donner une entête au rapport.
 - De `bottom title` permet de donner un pied de page au rapport
4. `5.`, `6.` et `7` `column`
 - Permet de changer l'entête d'une colonne utilisée dans les requêtes

`format` -> formatage dans le type

`word_wrapped` -> envoie à la ligne les caractères excédant le formatage

`truncate` -> élimine les caractères excédant le formatage

`heading` -> permet de renommer une colonne

Description de `activite.sql`

8. `break on`

- Pour rajouter une ligne vide avant une valeur différent (dans ce cas de Name) :

```
break on Name skip 1
```

- Par défaut on ne duplique pas les valeurs, si on veut les dupliquer faire :

```
break on Name duplicate skip 1
```

- Pour rajouter un total ou sous total

```
break on report
```

- Les deux commandes doivent être ensembles donc :

```
break on Name Skip 1 on report
```

- Doit être coordonné avec un `order by`, dans l'exemple :

```
order by Name
```

Description de activite.sql

9. compute avg

- Marche avec la commande break on, rajoute des astérisques et le mot avg

```
break on Name Skip 1 on report  
compute avg of DaysOut on Name
```

- On peut avoir toutes les fonctions de groupage de Oracle

```
compute sum  
compute count  
compute max  
...
```

Description de `activite.sql`

10. Set linesize

Set pagesize

Set newpage

`set feedback off | on | n` : évite les réactions de sqlplus après l'exécution des requêtes (e.g., `15 rows selected, table created`)

11. Spool

- Permet de démarrer l'envoi du contenu du rapport vers un fichier de type texte
`spool mon_fichier.lst`
- Doit être finalisé par
`spool off`
- Le fichier donné peut un nouveau fichier ou un fichier existant (par défaut un remplacement est fait)
`spool append mon_fichier.lst`
`spool replace mon_fichier.lst`

Configuration de votre environnement sqlplus

- Certaines commandes utilisées pour les rapports peuvent être utilisées pour configurer votre environnement sqlplus
- Configuration à définir dans le fichier `login.sql` à enregistrer dans votre répertoire de travail
- Exemple de fichier `login.sql`

```
set pagesize 100
set feedback 5
set sqlprompt 'You are the best > '
define _editor="emacs"
```

Miscellaneous

- Pour vérifier l'environnement de sqlplus
 - Pour certaines commandes il suffit de taper leur nom
 - `ttitle`
 - `btitle`
 - `column` (touts les formatages des colonnes seront montrés)
 - `break`
 - `compute`
 - Pour les commandes qui utilisent le mot clé `set` il faut utiliser le mot clé `show`
 - `Show pagesize`
 - `Show linesize`
 - `Show headsep`
 - `Show newpage`

Désactivation des commandes

- Pour désactiver

- `ttitle off`
- `btitle off`
- `clear columns`
- `clear breaks`
- `clear computes`
- `set pagesize`
- `set newpage`

- Certaines commandes doivent avoir toujours une valeur

- `set linesize 80`
- `set headsep |`

Fonctions pour obtenir de l'information sur les chaînes de caractères

Chaîne de caractères

- Une chaîne de caractères est composé de caractères alphanumériques (alphabet, numéros, signes de ponctuation et espaces blancs)
- Oracle gère deux types de chaînes de caractères
 - CHAR(max)
 - Les chaînes doivent être de la même longueur (max)
 - Si la longueur est variable alors on concatène de espaces blancs pour arriver à max
 - Longueur par défaut = 1
 - VARCHAR2(max)
 - Comme CHAR sauf que on ne rajoute pas de espaces blancs si la taille maximum n'est pas utilisée

Il est possible d'utiliser le type VARCHAR mais cela est déconseillé.

Fonctions pour les chaînes de caractères

Function Name	Use
	Glues or concatenates two strings together. The symbol is called a <i>vertical bar</i> or <i>pipe</i> .
ASCII	Returns the decimal representation in the database character set of the first character of the string.
CHR	Returns the character having the binary equivalent to the string in either the database character set or the national character set.
CONCAT	Concatenates two strings together (same as).
INITCAP	Initial capital. Capitalizes the first letter of a word or series of words.
INSTR	Finds the location of a character in a string.
LENGTH	Tells the length of a string.
LOWER	Converts every letter in a string to lowercase.
LPAD	Left pad. Makes a string a certain length by adding a certain set of characters to the left.
LTRIM	Left trim. Trims all the occurrences of any one of a set of characters off the left side of a string.
NLS_INITCAP	Initcap based on the National Language Support (NLS) value.
NLS_LOWER	Lower based on the NLS value.
NLS_UPPER	Upper based on the NLS value.
NLSSORT	Sort based on the national language selected.
REGEXP_INSTR, REGEXP_REPLACE, and REGEXP_ SUBSTR	INSTR, REPLACE, and SUBSTR for regular expressions.

Fonctions pour les chaînes de caractères

Function Name	Use
RPAD	Right pad. Makes a string a certain length by adding a certain set of characters to the right.
RTRIM	Right trim. Trims all the occurrences of any one of a set of characters off the right side of a string.
SOUNDEX	Finds words that sound like the example specified.
SUBSTR	Substring. Clips out a piece of a string.
TREAT	Changes the declared type of an expression.
TRIM	Trims all occurrences of any one of a set of characters off either or both sides of a string.
UPPER	Converts every letter in a string into uppercase.

Quelques exemples d'utilisation

- UPPER()
- Concaténation ||

```
select UPPER(nom), labo||' at ' || pays
from robots;
```

```
UPPER( LABO||'AT' ||PAYS
```

```
-----
BOB      TIM at USA
GASTON SAAL at France
ZOE      AIRNI at France
POK      SISCAN at Japan
TOM      GRRL at Germany
TOTO     TIM at USA
PIKPIK   SISCAN at Japan
PYROLI   GRRL at Germany
JOINEX   BIDON at Japan
```

- LOWER()
- concat(string1, string2)

```
select concat(LOWER(labo),' at '), pays
from robots;
```

```
CONCAT(LOW PAYS
```

```
-----
tim at      USA
saal at     France
airni at    France
siscan at   Japan
grrl at     Germany
tim at      USA
siscan at   Japan
grrl at     Germany
bidon at    Japan
```

Fonctions

- LPAD, RPAD, LTRIM, RTRIM, TRIM, LENGTH, SUBSTR et INSTR
- Chacune de ces fonctions a un objectif particulier
- Elles peuvent être utilisées ensemble

Quelques exemples d'utilisation

■ RPAD

```
select distinct(rpad(titre,15,'.')),demi_j  
from epreuves;
```

```
(RPAD(TITRE,15,DE  
----- --  
Construct..... AM  
Course1..... AM  
Course1b..... AM  
Course2..... PM  
Football..... AM  
Le combat1..... PM  
Le combat2..... PM  
Le jardin..... AM  
Le tennis..... AM  
Le tennis..... PM  
Planif..... PM  
Pugilat..... PM  
Toutterrain.... AM
```

■ LPAD

```
select distinct(lpad(titre,15)),demi_j  
from epreuves;
```

```
(LPAD(TITRE,15) DE  
----- --  
Planif PM  
Course1 AM  
Course2 PM  
Pugilat PM  
Course1b AM  
Football AM  
Construct AM  
Le jardin AM  
Le tennis AM  
Le tennis PM  
Le combat1 PM  
Le combat2 PM  
Toutterrain AM
```

Quelques exemples d'utilisation

- TRIM, LTRIM et RTRIM

- Considérez une table magazine avec un attribut TITLE contenant de valeurs pas très homogènes

```
select Title from MAGAZINE;
```

```
TITLE
-----
THE BARBERS WHO SHAVE THEMSELVES.
"HUNTING THOREAU IN NEW HAMPSHIRE"
THE ETHNIC NEIGHBORHOOD
RELATIONAL DESIGN AND ENTHALPY
"INTERCONTINENTAL RELATIONS."
```

- Ces fonctions permettent d'homogénéiser les chaînes de caractères à la sortie (pas de mise à jour sur la BD)

```
select RTRIM(Title, '.') from MAGAZINE
```

```
RTRIM(TITLE, '.')
-----
THE BARBERS WHO SHAVE THEMSELVES
"HUNTING THOREAU IN NEW HAMPSHIRE
THE ETHNIC NEIGHBORHOOD
RELATIONAL DESIGN AND ENTHALPY
"INTERCONTINENTAL RELATIONS
```

```
select TRIM(' ' from Title) from MAGAZINE;
```

```
TRIM(' ' FROM TITLE)
-----
THE BARBERS WHO SHAVE THEMSELVES.
HUNTING THOREAU IN NEW HAMPSHIRE
THE ETHNIC NEIGHBORHOOD
RELATIONAL DESIGN AND ENTHALPY
INTERCONTINENTAL RELATIONS.
```

```
select LTRIM(RTRIM(Title, '.'), '') from MAGAZINE
```

```
LTRIM(RTRIM(TITLE, '.'), '')
-----
THE BARBERS WHO SHAVE THEMSELVES
HUNTING THOREAU IN NEW HAMPSHIRE
THE ETHNIC NEIGHBORHOOD
RELATIONAL DESIGN AND ENTHALPY
INTERCONTINENTAL RELATIONS
```

Quelques exemples d'utilisation

- LENGTH(string), SUBSTR(string, start [, count])

```
select titre, SUBSTR(titre,1,4), LENGTH(titre)
from epreuves;
```

TITRE	SUBS	LENGTH(TITRE)
Le tennis	Le t	9
Le jardin	Le j	9
Le combat1	Le c	10
Le combat2	Le c	10
Toutterrain	Tout	11
Planif	Plan	6
Construct	Cons	9
Pugilat	Pugi	7
Course1	Cour	7
Course2	Cour	7
Course1b	Cour	8
Football	Foot	8
Le tennis	Le t	9
Toutterrain	Tout	11
Planif	Plan	6

```
select titre, SUBSTR(titre,5)
from epreuves;
```

TITRE	SUBSTR(T
Le tennis	ennis
Le jardin	ardin
Le combat1	ombat1
Le combat2	ombat2
Toutterrain	terrain
Planif	if
Construct	truct
Pugilat	lat
Course1	se1
Course2	se2
Course1b	selb
Football	ball
Le tennis	ennis
Toutterrain	terrain
Planif	if

Start negatif ?
Types CHAR ?

Ceci est équivalent à
select titre, SUBSTR(titre,5,8)
from epreuves;

Quelques exemples d'utilisation

■ INSTR(string, set [,start [,occurrence]])

```
select nom, instr(nom,'o')
from robots;
```

Est équivalent à :

```
select nom , instr(nom,'o',1,1)
from robots;
```

NOM	INSTR(NOM, 'O')
Bob	2
Gaston	5
Joinex	2
Pikpik	0
Pok	2
Pyroli	4
Tom	2
Toto	2
Zoe	2

```
select Author, INSTR(Author,'WILLIAM') from MAGAZINE;
```

AUTHOR	INSTR(AUTHOR, 'WILLIAM')
BONHOEFFER, DIETRICH	0
CHESTERTON, G.K.	0
RUTH, GEORGE HERMAN	0
WHITEHEAD, ALFRED	0
CROOKES, WILLIAM	10

```
select Author, SUBSTR(Author,1,INSTR(Author,',')-1)
from MAGAZINE;
```

AUTHOR	SUBSTR(AUTHOR, 1, INSTR(AUT
BONHOEFFER, DIETRICH	BONHOEFFER
CHESTERTON, G.K.	CHESTERTON
RUTH, GEORGE HERMAN	RUTH
WHITEHEAD, ALFRED	WHITEHEAD
CROOKES, WILLIAM	CROOKES

```
select Author, SUBSTR(Author,INSTR(Author,',')+2) from MAGAZINE;
```

AUTHOR	SUBSTR(AUTHOR, INSTR(AUTHO
BONHOEFFER, DIETRICH	DIETRICH
CHESTERTON, G.K.	G.K.
RUTH, GEORGE HERMAN	GEORGE HERMAN
WHITEHEAD, ALFRED	ALFRED
CROOKES, WILLIAM	WILLIAM

Quelques exemples d'utilisation

■ LENGTH

```
select nom
from robots
where length(nom)>3;
```

```
NOM
-----
Gaston
Joinex
Pikpik
Pyroli
Toto
```

```
select nom
from robots
order by length(nom);
```

```
NOM
-----
Bob
Pok
Tom
Zoe
Toto
Gaston
Pyroli
Joinex
Pikpik
```

■ SOUNDEX(string)

Cherche de chaînes de caractère dont la prononciation ressemble à la chaîne de caractères donnée

```
select nom
from robots
where soundex(nom)=soundex('picpic');
```

```
NOM
-----
Pikpik
```

```
select nom
from robots
where soundex(nom)=soundex('piroli');
```

```
NOM
-----
Pyroli
```

```
select nom
from robots
where soundex(nom)=soundex('bop');
```

```
NOM
-----
Bob
```

Quelques exemples d'utilisation

- REPLACE(char, search_string [,replace_string])

```
Select replace('Introduction a la physique',  
             'physique', 'chimie') as replace  
from dual;
```

REPLACE

Introduction a la chimie

```
Select replace('Introduction a la physique',  
             'physique') as replace  
from dual;
```

REPLACE

Introduction a la

```
select replace(nom,nom,'Mr '|nom) robot  
from robots;
```

ROBOT

Mr Bob
Mr Gaston
Mr Joinex
Mr Pikpik
Mr Pok
Mr Pyroli
Mr Tom
Mr Toto
Mr Zoe

Quelques exemples d'utilisation

```
Select replace('EGOREG','EG','GE')
from dual;
```

```
REPLAC
-----
GEORGE
```

Combien de remplacements ont été faits ?

```
select (length('EGOREG')-length(replace('EGOREG','EG',null)))
       /length('EG') as counter
from dual;
```

```
COUNTER
-----
                2
```

Recherche d'expressions régulières

Expressions régulières

- Les expressions rationnelles ou expressions régulières (en anglais *regular expressions* dont l'abrégé est **regexp** ou **regex**) sont une famille de notations compactes et puissantes pour décrire certains ensembles de chaînes de caractères.

Fonctions pour les expressions régulières

- Permettent de parcourir de chaînes de caractères à la recherche de motifs pour les récupérer, extraire, remplacer, etc.
- Dans Oracle 10g, les fonctions SUBSTR, INSTR, LIKE et REPLACE ont été améliorées
 - REGEXP_SUBSTR
 - REGEXP_INSTR
 - REGEXP_LIKE
 - REGEXP_REPLACE

Symboles

- Les expressions régulières sont composées de symboles (caractères et métacaractères)
- Les métacaractères
 - `^ . [] $ () * + ? | { } \-`
- Caractère littéral
 - Valeur qui est écrite exactement comme elle est interprétée
- Les symboles de début et fin de chaîne et le point
 - `^` Indique le début de la chaîne - exemple `^chat`
 - `$` Indique la fin de la chaîne - exemple : `chat$`
 - `.` Le point indique n'importe quel caractère

Symboles

- Les symboles quantificateurs
 - * Indique 0, 1 ou plusieurs occurrences du caractère ou de la classe précédente
 - + Indique une ou plusieurs occurrences du caractère ou de la classe précédente
 - ? Indique 0 ou une occurrence du caractère ou de la classe précédente
- Les intervalles de reconnaissance
 - $a\{3\}$ correspond exactement à aaa
 - $a\{2,\}$ correspond à un minimum de deux a consécutifs soit aa, aaa, aaaa....
 - $a\{2,4\}$ correspond uniquement à aa, aaa, aaaa
- Les classes de caractères
 - $br[iu]n$ trouver br suivi de i ou de u suivi de n

Symboles

■ L'intervalle

- [0-9] tous les chiffres de 0 à 9

Le tiré – peut être utilisé comme littéral uniquement en début d'expression

■ L'alternative

- p(ai|i)n tout ce qui s'écrit pain ou pin
- ^(De|A):@ tout ce qui commence par De:@ ou A:@

■ La classe complémentée

- [^1] tout sauf le chiffre 1
- [^1-6] tout sauf les chiffres de 1 à 6

Symboles

■ Les classes prédéfinies

- `[:alpha:]` n'importe quelle lettre
- `[:digit:]` n'importe quel chiffre
- `[:xdigit:]` caractères hexadécimaux
- `[:alnum:]` n'importe quelle lettre ou chiffre
- `[:space:]` n'importe quel espace blanc
- `[:punct:]` n'importe quel signe de ponctuation
- `[:lower:]` n'importe quelle lettre en minuscule
- `[:upper:]` n'importe quelle lettre capitale
- `[:blank:]` espace ou tabulation
- `[:graph:]` caractères affichables et imprimables
- `[:cntrl:]` caractères d'échappement
- `[:print:]` caractères imprimables exceptés ceux de contrôle

Operator	Description
----------	-------------

\ ^a	The backslash character can have four different meanings, depending on the context. It can stand for itself, quote the next character, introduce an operator, or do nothing.
*	Matches zero or more occurrences.
+	Matches one or more occurrences.
?	Matches zero or one occurrence.
	Alternation operator for specifying alternative matches.
^ ^b	Matches the beginning-of-line character.
\$ ^b	Matches the end-of-line character.
. ^c	Matches any character in the supported character set except NULL .
[] ^d	Bracket expression for specifying a matching list that should match any one of the expressions represented in the list. A nonmatching list expression begins with a caret (^) and specifies a list that matches any character except for the expressions represented in the list.
()	Grouping expression, treated as a single subexpression.
{ <i>m</i> }	Matches exactly <i>m</i> times.
{ <i>m</i> ,}	Matches at least <i>m</i> times.
{ <i>m</i> , <i>n</i> }	Matches at least <i>m</i> times but no more than <i>n</i> times.
\ <i>n</i> ^e	The backreference expression (<i>n</i> is a digit between 1 and 9) matches the <i>n</i> th subexpression enclosed between parentheses and preceding \ <i>n</i> .
[.. ^f	Specifies one collation element and can be a multicharacter element (for example, [.ch.] in Spanish).
[: :] ^g	Specifies character classes (for example, [:alpha:]). It matches any character within the character class.
[= =] ^h	Specifies equivalence classes. For example, [=a=] matches all characters having base letter 'a'.

Notes on the POSIX operators and Oracle enhancements:

- a. The backslash operator can be used to make the character following it normal if it is an operator. For example, '*' is interpreted as the asterisk string literal.
- b. The characters '^' and '\$' are the POSIX anchoring operators. By default, they match only the beginning or end of an entire string. Oracle lets you specify '^' and '\$' to match the start or end of any line anywhere within the source string. This, in turn, lets you treat the source string as multiple lines.
- c. In the POSIX standard, the “match any character” operator (.) is defined to match any English character except **NULL** and the newline character. In the Oracle implementation, the '.' operator can match any character in the database character set, including the newline character.
- d. In the POSIX standard, a range in a regular expression includes all collation elements between the start and end points of the range in the linguistic definition of the current locale. Therefore, ranges in regular expressions are linguistic ranges rather than byte values ranges, and the semantics of the range expression are independent of character set. Oracle implements this independence by interpreting range expressions according to the linguistic definition determined by the **NLS_SORT** initialization parameter.
- e. The backreference expression '\n' matches the same string of characters as was matched by the *n*th subexpression. The character *n* must be a digit from 1 to 9, designating the *n*th subexpression, numbered from left to right. The expression is invalid if the source string contains fewer than *n* subexpressions preceding the \n. For example, the regular expression ^(.*)\1\$ matches a line consisting of two adjacent appearances of the same string. Oracle supports the backreference expression in the regular expression pattern and the replacement string of the **REGEXP_REPLACE** function.
- f. A collating element is a unit of collation and is equal to one character in most cases, but may comprise two or more characters in some languages. Historically, regular expression syntax does not support ranges containing multicharacter collation elements, such as the range 'a' through 'ch'. The POSIX standard introduces the collation element delimiter '[..]', which lets you delimit multicharacter collection elements such as '[a-[.ch.]]'. The collation elements supported by Oracle are determined by the setting of the **NLS_SORT** initialization parameter. The collation element is valid only inside the bracketed expression.
- g. In English regular expressions, range expressions often indicate a character class. For example, '[a-z]' indicates any lowercase character. This convention is not useful in multilingual environments where the first and last character of a given character class may not be the same in all languages. The POSIX standard introduces the portable character class syntax '[::]'.
- h. Oracle supports the equivalence classes through the POSIX '[==]' syntax. A base letter and all of its accented versions constitute an equivalence class. For example, the equivalence class '[=a=]' matches ä and â. The equivalence classes are valid only inside the bracketed expression.

Fonction REGEXP_SUBSTR

- Extrait une partie d'une chaîne de caractères

```
REGEXP_SUBSTR(source_string, pattern
              [, position (Par défaut le premier caractère)
              [, occurrence (Par défaut 1)
              [, match_parameter ] ('i'|'c')
              ]
              )
```

→ 'i' case-insensitive
'c' case sensitive
Par défaut 'c'

```
Select REGEXP_SUBSTR ('MY LEDGER: Debits, Credits, and Invoices 1940','my`
      , 1, 1, 'i') "Regexp_substr"
```

```
From dual;
```

```
Re
```

```
--
```

```
MY
```

```
SELECT REGEXP_SUBSTR('first field, second field , third field',' ', [^,]*,')
from dual;
```

```
REGEXP_SUBSTR('F
```

```
-----
```

```
, second field ,
```

Fonction REGEXP_SUBSTR

```
select regexp_substr('123-456-789','-[^-]')  
from dual;
```

```
RE  
--  
-4
```

```
select regexp_substr('123-456-789','-[^-]+',1,1)  
from dual;
```

```
REGE  
----  
-456
```

```
select regexp_substr('123-456-789','-[^-]+',5,1)  
from dual;
```

```
REGE  
----  
-789
```

```
select regexp_substr('123-456-789','-[^-]+',1,2)  
from dual;
```

```
REGE  
----  
-789
```

Fonction REGEXP_INSTR

- Donne la position initiale d'un motif

```
REGEXP_INSTR (source_string, pattern
              [, position   (Par défaut le premier caractère)
              [, occurrence (Par défaut 1)
              [, return_option (1|0)
              [, match_parameter ] ('|c')
              ]
              ]
              )
```

- 1 donne la position du caractère suivant
- 0 donne la position du caractère qui coïncide avec le motif
- Par défaut 0

```
Select regexp_instr('2, rue de la Houssinière BP 92208, 44322 Nantes','[[:digit:]]{5}',
  2 1,1,1) as regexp_instr from dual;
REGEXP_INSTR
-----
          34
```

```
SQL> Select regexp_instr('2, rue de la Houssinière BP 92208, 44322 Nantes','[[:digit:]]{5}',
  2 1,1,0) as regexp_instr from dual;
REGEXP_INSTR
-----
          29
```

```
Select regexp_instr('2, rue de la Houssinière BP 92208, 44322 Nantes','[[:digit:]]{5}',
  2 10,2) as regexp_instr from dual;
REGEXP_INSTR
-----
          36
```

Fonction REGEXP_INSTR

```
SELECT REGEXP_INSTR('Joe Smith, 10045 Berry Lane, San Joseph, CA 91234',
                    '[:digit:]{5}$') AS rx_instr
FROM dual;
```

```
RX_INSTR
-----
         45
```

```
select nom, regexp_instr(nom,'e$|^P') pos
from robots;
```

NOM	POS
Bob	0
Gaston	0
Joinex	0
Pikpik	1
Pok	1
Pyroli	1
Tom	0
Toto	0
Zoe	3

```
select nom, regexp_instr(nom,'e$|o|^P') pos
from robots;
```

NOM	POS
Bob	2
Gaston	5
Joinex	2
Pikpik	1
Pok	1
Pyroli	1
Tom	2
Toto	2
Zoe	2

Fonction REGEXP_LIKE

- Recherche un motif dans une chaîne de caractères
- Elle est utilisée dans le WHERE d'une requête

```
REGEXP_LIKE (source_string, pattern
             [match_parameter ]
             )
```

```
select *
from robots
where regexp_like (pays, 'Fra+');
NOM      LABO      PAYS      CAT
-----
Gaston SAAL      France    1
Zoe      AIRNI     France    1
```

```
select *
from robots
where regexp_like (pays, 'fra+', 'i');
NOM      LABO      PAYS      CAT
-----
Gaston SAAL      France    1
Zoe      AIRNI     France    1
```

```
select *
from robots
where regexp_like (pays, 'fra+', 'c');

no rows selected
```

```
select *
from robots
where regexp_like (nom, 'B.b');
NOM      LABO      PAYS      CAT
-----
Bob      TIM      USA      1
```

```
select *
from robots
where regexp_like (nom, '^P|b$');
NOM      LABO      PAYS      CAT
-----
Bob      TIM      USA      1
Pok      SISCAN   Japan    2
Pikpik   SISCAN   Japan    1
Pyroli   GRRL     Germany  2
```

Fonction REGEXP_REPLACE

- Remplace une chaîne de caractères dans un string d'après un motif

```
REGEXP_REPLACE(source_string, pattern
               [, replace_string
               [, position
               [, occurrence
               [, match_parameter ]
               ]
               ]
               ]
               )
```

```
SELECT REGEXP_REPLACE('Joe   Smith','( ){2,}',' ')
       AS RX_REPLACE
FROM dual
RX_REPLACE
-----
Joe Smith
```

Backreferences \n

- Permet de stocker de sous expressions pour une future utilisation
- Chaque sous expression est numérotée du 0 à 9 dans l'ordre d'apparition

```
SELECT REGEXP_REPLACE('025-112-5812','0([[[:digit:]]{2})-([[[:digit:]]{3})-([[[:digit:]]{4})','+33 (0)\1 \2-\3') REGEXP_REPLACE
```

```
FROM dual;
```

```
REGEXP_REPLACE
```

```
-----  
+33 (0)25 112-5812
```

```
SELECT REGEXP_REPLACE('Ellen Hildi Smith','(.*) (.*) (.*)', '\3, \1 \2')  
NOM_PRENOM FROM dual;
```

```
NOM_PRENOM
```

```
-----  
Smith, Ellen Hildi
```

Expressions REGEXP dans les contraintes

- La création de la table suivante a comme contraintes
 - Le nom et le prénom auront uniquement de caractères alphabétiques
 - Le code_postal aura 5 chiffres
 - La carte_credit aura 4 * 4 chiffres séparés par un espace ou signe de ponctuation ou rien.

```
CREATE TABLE CLIENTS(  
NOM varchar2(30) check (REGEXP_LIKE(NOM, '[:alpha:]+$')),  
PRENOM varchar2(30) check (REGEXP_LIKE(prenom, '[:alpha:]+$')),  
CODE_POSTAL int check (REGEXP_LIKE(CODE_POSTAL, '[:digit:]{5}')),  
CARTE_CREDIT varchar2(30) check (REGEXP_LIKE(CARTE_CREDIT,  
  '(([:digit:]{4})([:space:]|[:punct:])* ){4}')));
```

Références

- http://download-uk.oracle.com/docs/cd/B14117_01/index.htm
- http://www.oracle.com/technology/oramag/webcolumns/2003/techarticles/rischert_regexp_pt1.html
- <http://www.sciences.univ-nantes.fr/info/perso/permanents/Patricia.Serrano-Alvarado/Oracle10gTheReference.pdf>
- http://www.sciences.univ-nantes.fr/info/perso/permanents/Patricia.Serrano-Alvarado/TWP_Regular_Expressions.pdf