



# Mémo SQL

\* => nouveauté 7.3

## Ordres de base

SQL 89

```

SELECT colonne1 [as entete1],
           colonne2 [as entete2]
( select imbriqué retournant une valeur) as entete3
FROM fichier1 [f1], fichier2 [f2]
WHERE [critères de jointure et sélection]
GROUP BY colonne
HAVING [sélection]

ORDER BY colonne [ASC|DESC]
ou N°-de-colonne

FETCH FIRST n ROWS ONLY

```

SQL 92

```

SELECT colonne1 [as entete1],
           colonne2 [as entete2]
( select imbriqué retournant une valeur) as entete3
FROM fichier1 f1 join1 fichier2 f2 ON f1.clea = f2.clea
                                     and f1.cleb = f2.cleb
                                     [join fichier3 f3 on f2.clec = f3.clec]

```

<sup>1</sup> options de jointure

<b>Join</b>	<i>uniquement les enregistrements en correspondance</i>
<b>Left outer join</b>	<i>tous les enregistrements de fichier1</i>
<b>exception join</b>	<i>uniquement les enregistrements sans correspondance</i>
<b>right outer join</b>	<i>tous les enregistrements de fichier2</i>
<b>right exception join</b>	<i>uniquement les enregistrements sans correspondance</i>
<b>full outer join</b>	<i>toutes les combinaisons (right et left outer)</i>

```

WHERE [sélection]
GROUP BY [CUBE | ROLLUP] colonne (ou expression)
           HAVING [sélection]

ORDER BY colonne [ASC|DESC]
           ou N°-de-colonne (Limit) 2

```

	<sup>2</sup> options de limitation  <b>FETCH</b> FIRST n ROWS ONLY (affiche les n premières lignes) ou <b>LIMIT</b> n OFFSET Y (affiche les n premières lignes à partir de y+1)	
(tables dérivées)	<i>Soit</i> <b>SELECT</b> colonne1 [as entete1],colonne2 [as entete2] <b>FROM</b> (SELECT ... FROM ...) as nom-temporaire <b>WHERE</b> [sélection]  <i>Soit</i> <b>WITH</b> nom-temporaire as (SELECT ... FROM ...) <b>SELECT</b> colonne1 [as entete1],colonne2 [as entete2] <b>FROM</b> nom-temporaire <b>WHERE</b> [sélection]	
(Requête récursive)	<b>WITH temp</b> (composant, compose, quantite) as (select L.composant, L.compose, L.quantite from liens L where compose = 'voiture' UNION ALL select fils.composant , fils.compose, fils.quantite <b>from temp</b> AS Pere join liens AS Fils on pere.compose=Fils.composant ) <b>SELECT * FROM TEMP</b>	
(Requête récursive ou hiérarchique V7)	SELECT chef, matricule, nom FROM personnel <b>START WITH</b> chef = 'Mr le Directeur' <b>CONNECT BY PRIOR</b> matricule =chef  options liées :	
	<b>ORDER SIBLINGS</b> BY nom	Tri les lignes ayant le même parent
	CONNECT BY <b>NOCYCLE</b> PRIOR	évite l'arrêt en erreur lors d'une boucle infinie (la ligne qui provoque la boucle est affichée une deuxième fois)
	<b>CONNECT_BY_ISCYCLE</b>	retourne 1 si la ligne en cours aurait provoqué une boucle
	<b>CONNECT_BY_ISLEAF</b>	retourne 1 si la ligne en cours n'a pas d'enfant
	<b>LEVEL</b>	indique le niveau dans la hiérarchie pour cette ligne
	<b>CONNECT_BY_ROOT</b>	indique l'élément racine (le parent d'origine) pour cette ligne
<b>SYS_CONNECT_BY_PATH</b> ( <élément> , <séparateur>)	retourne le chemin complet (suite de <élément> séparés par <séparateur>)  par exemple : SELECT SYS_CONNECT_BY_PATH(trim(chef), '/') AS chemin retourne sous forme de <u>CLOB</u> :  /Mr le Directeur /Mr le Directeur/Michelle /Mr le Directeur/Michelle/Françoise /Mr le Directeur/Michelle/Françoise/Yves	

(Tables * temporelles)	-- comme si nous étions le 10 Février 2016 <b>Select * From fichier</b> <b>for system_time as of '2016-02-10-12.00.00.00000000000'</b> ou <b>Select * From fichier</b> <b>from '2016-02-01-00.00.00.00000000000'</b> <b>to '2016-02-10-23.59.59.00000000000'</b>
UPDATE (-> V4R20)	<b>UPDATE</b> fichier <b>SET</b> colonnel = valeur1 <b>WHERE</b> [sélection]
UPDATE (V4R30 et +)	<b>UPDATE</b> fichier f <b>SET</b> colonnel = (select valeur1 from autrefichier where cle = f.cle) <b>WHERE</b> [sélection] <b>Order</b> BY x <b>LIMIT</b> n <b>OFFSET</b> y *
INSERT	<b>INSERT INTO</b> fichier <b>VALUES</b> (valeur1, touteslescolonnes...) ou <b>INSERT INTO</b> fichier (colonne2, colonne3) <b>VALUES</b> (v2, v3) ou <b>INSERT INTO</b> fichier ( <b>SELECT ... FROM ...WHERE ...</b> )
INSERT + FINAL TABLE	<b>SELECT</b> cle_auto, quand <b>FROM FINAL TABLE</b> ( <b>INSERT INTO</b> fournisseur (raisoc, quand) <b>VALUES</b> ('IBM', now() )) -- affiche la nouvelle ligne insérée
DELETE	<b>DELETE FROM</b> fichier <b>WHERE</b> [sélection] <b>Order</b> BY x <b>LIMIT</b> n <b>OFFSET</b> y *
TRUNCATE (CLRPFM)	<b>TRUNCATE TABLE</b> CLIENTP1 <b>DROP STORAGE</b> <b>IGNORE DELETE TRIGGERS RESTART IDENTITY</b>
VALUES	<b>VALUES</b> now() -- affiche le timestamp en cours
MERGE	<b>MERGE INTO</b> cible C <b>USING</b> (SELECT source1 , source2 FROM source) S <b>ON</b> (C.zonecle = S.zonecle) <b>WHEN MATCHED THEN</b> UPDATE SET cible2 = source2 <b>WHEN NOT MATCHED THEN</b> INSERT (cible1, cible2) <b>VALUES</b> (source1, source2)
Concurrence d'accès pour SELECT, UPDATE, DELETE et MERGE	<b>Suivant le niveau de commitment control</b> - <b>WAIT FOR OUTCOME</b> Attendre que les lignes verrouillées soient libérées (CS, ou RS) - <b>SKIP LOCKED DATA</b> les lignes verrouillées sont ignorées (NC, UR, CS, ou RS) - <b>USE CURRENTLY COMMITTED</b> Utiliser les (anciennes) valeurs déjà validées (SELECT sous CS uniquement)

## Sélections

Opérateur logique	Exemple(s)
<div style="display: flex; align-items: center;"> <div style="flex: 1;"> <p>colonne <b>op.</b> colonne ou colonne <b>op.</b> valeur</p> </div> <div style="border-left: 1px solid black; border-right: 1px solid black; padding: 0 5px; text-align: center;"> <p><b>op.</b></p> <p>—</p> <p>=</p> <p>&lt;</p> <p>&gt;</p> <p>&lt;&gt;</p> <p>&lt;=</p> <p>&gt;=</p> </div> </div>	<pre>QTECDE &lt;&gt; 0 LIBART = 'Pamplemousse' PRIX &gt;= 45</pre>
<b>IN</b> (val1, val2, val3, ...)	DEPT IN (44, 49, 22, 56, 29)
<b>BETWEEN</b> val1 <b>AND</b> val2	DEPT BETWEEN 44 AND 85
<b>LIKE</b>	<pre>nom LIKE 'DU%' (commence par) nom LIKE '%PON%' (contient) nom LIKE '%RAND' (se termine par)  <i>depuis la V5R1 : nom LIKE '%concat ville concat '%'</i></pre>
<b>IS (IS NOT) NULL</b>	test la valeur nulle ( <i>pratique avec les jointures externes</i> )
et aussi <b>OR, AND, NOT, (, )</b> .	CODART = 1245 or LIBART = 'Pamplemousse'
<b>V5R40</b> : les valeurs peuvent être comparées en ligne	... where (cepage1, cepage2) = ('Syrah' , 'Grenache')
<b>REGEXP_LIKE</b> (zone, <expression régulière>)	Where REGEXP_LIKE(pr_nom , 'ch[aâä]teau')
<b>IS (IS NOT) JSON *</b>	Where JDATA IS JSON
<b>JSON_EXISTS *</b>	Where JSON_EXISTS(JDATA 'strict \$.TEL' FALSE ON ERROR)

## Fonctions valides (ligne à ligne ou scalaires)

Fonction(x)	Retourne ?	Exemple
MAX(X,Y)	retourne la plus grande valeur de X ou de Y	MAX(prixHA, pritarif) * qte
MIN(X,Y)	retourne la plus petite valeur de X ou de Y	MIN(datce, datliv)
ABSVAL(x)	la valeur absolue de x	ABSVAL(prix) * qte
CEIL(x)	Retourne l'entier immédiatement supérieur à X	CEIL(2,42) = 3 CEIL(2,56) = 3
FLOOR(x)	Retourne l'entier inférieur à X	FLOOR(2,42) = 2 FLOOR(2,56) = 2
RAND()	Retourne un nombre aléatoire	
ROUND(x , y)	Retourne l'arrondi comptable à la précision y	ROUND(2,42 , 1) = 2,40 ROUND(2,56 , 1) = 2,60
SIGN(x)	Retourne -1 si x est négatif, 1 s'il est positif, 0 s'il est null	Where SIGN(x) = -1
TRUNCATE(x , y)	Retourne le chiffre immédiatement inférieur à X (à la précision y)	TRUNCATE(2,42 , 1) = 2,40 TRUNCATE(2,56 , 1) = 2,50
DEC(x , l, d)	x au format numérique packé avec la lg et la précision demandée.	DEC(zonebinaire) DEC(avg(prix) , 9, 2)
DIGITS(x)	x en tant que chaîne de caractères	DIGITS(datnum)
CHAR(x)	x en tant que chaîne de car. (x étant une date)	CHAR(current date)
CHAR(V)	V en tant que CHAR (V étant un VARCHAR)	CHAR(zonevariable)

VARCHAR_FORMAT(X)	retourne en VARCHAR une chaîne CHAR	Select VARCHAR_FORMAT(PR_NOM) from producteurs
VARCHAR_FORMAT(X, 'unmasque')	retourne en chaîne une information numérique  <u>avec dans le masque :</u> 0 un chiffre 9 un chiffre à blanc si non significatif . le point comme marque décimale , la virgule comme séparateur de milliers S le signe (+ ou -) \$ le symbole monétaire \$ L le symbole monétaire en cours D la marque décimale en cours G le séparateur de groupe en cours	VALUES VARCHAR_FORMAT(1,5 - 1,4, '0D0') -> '0,1'
BLOB(x)	une chaîne de car. (x) en tant que BLOB	BLOB('texte de la chaîne à convertir')
FLOAT(x)	x au format "virgule flottante"	FLOAT(qte)
INT(x)	x au format binaire	INT(codart)
ZONED(x)	x au format numérique étendu	ZONED(prix)

CAST(x as typeSQL[lg])	<p>x au format indiqué par typeSQL :</p> <p>types valides :</p> <p>INT   INTEGER SMALLINT DEC(lg, dec) NUMERIC(lg, dec) FLOAT   REAL   DOUBLE CHAR   VARCHAR --FOR BIT DATA-- --FOR SBCS-- ----FOR n°-ccsid *-- DATE TIME TIMESTAMP</p> <p>*(FR = 297, US = 37)</p>	<p>CAST(qte AS CHAR(9)) <i>Attention les zéros de gauche sont éliminés</i></p> <p>CAST(prixchar as NUMERIC(7, 2))</p> <p>cast('123456,89' as numeric(8, 2)) <i>fonctionne</i></p> <p>cast('123456,89' as numeric(7, 2)) <i>donne une erreur (trop d'entiers)</i></p>
<p>STRIP ou TRIM(x [, 'c'])</p> <p>RTRIM(x [, 'c'])</p> <p>LTRIM(x [, 'c'])</p>	<p>supprime les blancs ou c aux deux extrémités</p> <p>blancs ou c de droite</p> <p>blancs ou c de gauche</p>	<p>TRIM(raisoc)</p>
<p>LENGTH(x) ou OCTET_LENGTH(x)</p>	<p>la longueur de x</p>	<p>LENGTH (nom)</p> <p>LENGTH (TRIM (nom) )</p>
CONCAT(x, y)	<p>concatene X et Y (<i>aussi</i> x CONCAT y <i>ou</i> X    Y)</p>	<p>CONCAT (nom, prenom)</p>
SUBSTR(x, d, l)	<p>extrait une partie de x depuis D sur L octets</p>	<p>SUBSTR (nom, 1, 10)</p> <p>SUBSTR (nom, length (nom), 1)</p>
LEFT(x, l)	<p>extrait une partie de x depuis 1 sur L octets</p>	<p>LEFT (nom, 10)</p>
RIGHT(x, l)	<p>extrait les L derniers octets de x</p>	<p>RIGHT (nom, 5)</p>
SPACE(n)	<p>retourne n blancs</p>	<p>nom concat space(5) concat prenom</p>
REPEAT(x, n)	<p>retourne n fois x</p>	<p>repeat ('*', 15)</p>
RPAD(chaine, n, 'c')	<p>Complète une chaîne à droite par le caractère indiqué (c) jusqu'à longueur de n</p>	<p>RPAD (PR_TEL, 15, '.')</p>

LPAD(chaine , n , 'c')	Complète une chaîne à gauche par le caractère indiqué (c) jusqu'à longueur de n	LPAD (PR_TEL, 15, '-')
MOD(x, y)	le reste de la division de x par y	MOD (annee, 4)
RRN(fichier)	N° de rang en décimal	RRN(clientp1)
RID(fichier)	N° de rang en binaire	RID(clientp1)
TRANSLATE(x) UPPER(x) UCASE(x)	X en majuscule	WHERE UCASE (RAISOC) LIKE 'VO%'
LOWER(x) LCASE(x)	x en minuscule	WHERE LCASE (ville) LIKE 'nan%'
TRANSLATE( x, remplace, origine)	Remplace tous les caractères de X présent dans <b>origine</b> par le caractère de même position dans <b>remplace</b> .	TRANSLATE (prixc, ' €', '0\$') <i>remplace 0 par espace et \$ par €</i>
REPLACE( x, origine, remplacement)	Remplace la chaîne de caractère <b>origine</b> par la chaîne <b>remplacement</b> dans x	REPLACE (x, 'Francs' , 'Euros') <i>--remplace <b>Francs</b> par <b>Euros</b></i>
REPLACE( x, origine) *	Supprime la chaîne de caractère <b>origine</b>	REPLACE (x, 'Francs' ) <i>--supprime <b>Francs</b></i>
SOUNDEX( x)	Retourne le SOUNDEX (représentation phonétique) de X [algorithme anglo-saxon]	Where SOUNDEX (prenom) = SOUNDEX ('Henri')
DIFFERENCE( x)	Retourne l'écart entre deux SOUNDEX	Where DIFFERENCE (prenom, 'HENRI') > 2 <i>--0 = très différents, 4 = très proches</i>
VALUE(x, y) IFNULL(x, y)	retourne X s'il est non null, sinon Y	IFNULL (DEPT, 0)
NULLIF(x, y)	retourne NULL si X = Y	NULLIF (Prix, 0)



LOCATE(x, y ,[d]) ou POSSTR(y , x)	retourne la position à laquelle x est présent dans y ou 0 (la recherche commence en <b>d</b> , qui est facultatif)	LOCATE(' ', raisoc)
POSITION(x IN y)	retourne la position à laquelle x est présent dans y ou 0	POSITION(' ' IN raisoc)
LOCATE_IN_STRING(y, x, d, n)	retourne la position de la <i>Nème</i> occurrence de x dans y à partir de D	LOCATE_IN_STRING('Bonjour', 'o', 1, 2) -> 5 --> 2ème o
INSERT(x, d, nb, ch)	insert <i>ch</i> dans x à la position <i>d</i> , en remplaçant <i>nb</i> octets ( <i>0 admis</i> )	
OVERLAY(x, ch, d, [nb])	insert <i>ch</i> dans x à la position <i>d</i> , en remplaçant <i>nb</i> octets ( <i>facultatif</i> )	OVERLAY('DB2 sur x est incroyable', 'IBMi', 9) -> 'DB2 sur IBMi ..'
DATABASE()	retourne le nom de la base (enregistré par WRKRDBDIRE)	
ENCRYPT_RC2(x, p, a)	Encrypte x en utilisant p comme mot de passe ( <i>a est l'astuce mémorisée</i> )	ENCRYPT_RC2(data, 'systemi', 'avant IBMi')
ENCRYPT_TDES(x, p, a)	Encrypte (algorithme TDES) x en utilisant p comme mot de passe ( <i>a est l'astuce mémorisée</i> )	ENCRYPT_TDES(data, 'systemi', 'avant IBMi')
ENCRYPT_AES(x, p, a)	Encrypte (algorithme AES) x en utilisant p comme mot de passe ( <i>a est l'astuce mémorisée</i> )	ENCRYPT_AES(data, 'systemi', 'avant IBMi')

GETHINT(x)	retrouve l'astuce associée à x	GET-HINT(data) --> 'avant IBMi'
DECRYPT_BIT(x, [pwd])	retourne (varchar) les données d'origine de x. (sans pwd, on doit lancer avant <i>SET ENCRYPTION PASSWORD = p</i> )	
DECRYPT_BINARY(x, [pwd])	retourne (en binary) les données d'origine de x.(cf <i>SET ENCRYPTION PASSWORD</i> )	
DECRYPT_CHAR(x, [pwd])	retourne (en VARCHAR) les données d'origine de x. (cf <i>SET ENCRYPTION PASSWORD</i> )	
VERIFY_GROUP_FOR_USER()	retourne 1 si l'utilisateur appartient à l'un des groupes indiqués, 0 sinon.	<pre> +---SESSION_USER---+ VERIFY_GROUP_FOR_USER(-+-----USER-----                         +-CURRENT_USER----+ +--, G1 [, Gn] --) </pre>
IDENTITY_VAL_LOCAL()	retourne la dernière valeur assigné à une zone avec l'attribut AS IDENTITY (voir plutôt la clause <a href="#">FINAL TABLE</a> , maintenant)	
HEX(chaine)	retourne la valeur hexa d'un chaine	<pre> HEX('bonjour') -&gt; 8296959196A499 X'8296959196A499' -&gt; Bonjour </pre>
VARBINARY_FORMAT()	retourne la chaine (FOR BIT DATA) d'une séquence binaire	<pre> VALUES VARBINARY_FORMAT('8296959196A499')-&gt; bonjour  varbinary_format('A7B37D20-915D-45C4- 8D0B-B5AA0F864FCA', </pre>

	(ayant le format indiqué, s'il est précisé)	'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX') - -adresse mac -> BX'A7B37D20915D45C48D0BB5AA0F864FCA (non affichable)
VARCHAR_FORMAT_BINARY()	retourne la valeur hexa, formatée, d'une chaîne.	VALUES VARCHAR_FORMAT_BINARY('123456789abcdef0', 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX')  -> F1F2F3F4-F5F6-F7F8-F981-8283848586F0
CASE when ... then .. when ... then .. [else] END	retourne la valeur du premier THEN ayant la clause WHEN de vérifiée.	CASE dept WHEN 44 then 'NANTES' WHEN 49 then 'ANGERS' ELSE 'Hors région' END AS METROPOLE  ou bien  CASE WHEN prix < 0 then 'négatif' WHEN codart = 0 then 'inconnu' ELSE 'positif ou nul' END

### Fonctions valides (fonctions de groupe ou d'agrégation)

Fonction(x)	Retourne ?	Exemple
MAX(X)	retourne la plus grande valeur sélectionnée	MAX (DATLIV)
MIN(X)	retourne la plus petite valeur sélectionnée	MIN (prix)
AVG(x)	la moyenne de X pour un groupe	AVG (quantite)
STDDEV(X)	retourne l'écart type	
VAR(X)	retourne la variance	
COUNT(*)	le nombre de lignes sélectionnées	
COUNT(X)	le nombre de lignes ou X est non null	COUNT (NOCLI)
COUNT(DISTINCT X)	le nombre de valeurs différentes rencontrées pour X	COUNT (distinct nocli)
SUM(x)	retourne la somme de X	SUM(qte * prix)

COVARIANCE(x , y) *	Indique si X et Y évoluent en même temps (en unités)	COVARIANCE (meteo, ventedeglace)
CORRELATION(x , y) *	Indique si X et Y évoluent en même temps (entre -1 et 1)	CORRELATION (PIB, GES)
MEDIAN(X) *	Valeur médiane de X	MEDIAN (salaire)
PERCENTILE_CONT(X) *	retourne le pourcentille : PERCENTILLE(0,5) = MEDIAN()	PERCENTILE_CONT (0,1) within group (order by SALAIRE) -- 1er décille
PERCENTILE_DISC(X) *	retourne le pourcentille discret (si nb de lignes paire -> retourne la 1ere valeur)	PERCENTILE_DISC (0,1) within group (order by SALAIRE)
REGR_COUNT(x , y) *	Droite de régression, nbr de paires non nulles	
REGR_INTERCEPT(x , y) *	Droite de régression, ordonnée à l'origine (valeur de x quand y =0)	
REGR_R2(x , y) *	Droite de régression, coefficient de détermination (écart moyen / droite)	REGR_R2(PIB, GES) // 1 = tous les points sur la droite
REGR_SLOPE(x , y) *	Droite de régression, pente de la droite théorique	REGR_SLOPE(PIB, qualiteAIR) //-1= pente descendante
REGR_AVGX(x , y) *	Droite de régression, moyenne de X sans les valeurs nulles	
REGR_AVGY(x , y) *	Droite de régression, moyenne de Y sans les valeurs nulles	
REGR_SXX(x , y) *	REGR_COUNT(X, Y) * VARIANCE(Y)	
REGR_SXY(x , y) *	REGR_COUNT(X, Y) * COVARIANCE(X, Y)	
REGR_SYY(x , y) *	REGR_COUNT(X, Y) * VARIANCE(X)	
LISTAGG ( x, 's') *	concatène toutes les occurrences de X (avec s comme séparateur)	SELECT pr_nom , <b>LISTAGG</b> (vin_nom, ',') WITHIN GROUP (order by vin_code) FROM producteur join vins Using(pr_code) <b>GROUP BY</b> pr_code, pr_nom

## Fonctions OLAP

ROW_NUMBER()	numérote les lignes affichées	select ROW_NUMBER() <b>over</b> (), codart, libart from articles [order by un tri]
	numérote les lignes sur un critère de tri (ici le prix)	select ROW_NUMBER() <b>over (order by prix)</b> , codart, libart from articles [order by autre-tri]
	numérote les lignes sur un tri (le prix) à l'intérieur d'une même famille	select ROW_NUMBER() <b>over (Partition by FAM Order by prix)</b> , codart, libart from articles [order by autre-tri]
RANK()	attribue un rang unique à chaque ligne, en gérant les ex-aequo (par exemple 1-1-3-4-4-4-7)	select RANK() <b>over ([Partition by..] order by prix)</b> , codart, libart from articles
DENSE_RANK()	attribue un rang unique et consécutif (sans trou) à chaque ligne (par exemple 1-1-2-3-3-3-4)	select DENSE_RANK() <b>over ([Partition by..] order by prix)</b> , codart, libart from articles
PERCENT_RANK()*	Retourne le % du rang (entre 0 et 1)	select PERCENT_RANK() <b>over ([Partition by..] order by prix)</b> , codart, libart from articles
LAG(X)*	Retourne la valeur de X de la ligne du dessus	select Annee, CA , (CA - LAG(CA) over (order by annee) ) as evolution FROM factures
LEAD(X)*	Retourne la valeur de X de la ligne du dessous	
NTILE(X)*	Retourne la quantile (NTILE(10) = le décile)	select <b>NTILE(3) over (order by cacli )</b> , CACLI ,NOMCLI FROM clients - par tiers

CUME_DIST(X)*	Retourne la distribution cumulée (le dernier vaut 1)	select <b>cume_dist()</b> over <b>(order by cacli )</b> , CACLI ,NOMCLI FROM clients
FIRST_VALUE(X)*	Retourne la première valeur de X suivant le tri	select CACLI , cacli / <b>first_value(cacli)</b> over (order by cacli) AS NBRDEFOISPLUS FROM clients
LAST_VALUE(X)*	Retourne la dernière valeur de X suivant le tri	select CACLI , cacli / <b>LAST_value(cacli)</b> over (order by cacli) AS NBRDEFOISMOINS FROM clients
NTH_VALUE(X, N)*	Retourne la Nième valeur de X suivant le tri	
RATIO_TO_REPORT(X)*	Retourne le % de la somme cumulée	select CACLI , RATIO_TO_REPORT ( <b>cacli</b> ) over (order by cacli) AS RATIO FROM clients

<p><u>V6 : options pour GROUP BY</u></p>	<p>soit le SELECT basique suivant -&gt;</p>	<p>SELECT SOC, DEP, Count(*) .../... GROUP BY soc, Dep</p> <table border="1" data-bbox="927 264 1158 629"> <thead> <tr> <th>SOC</th> <th>DEP</th> <th>Count(*)</th> </tr> </thead> <tbody> <tr> <td>01</td> <td>22</td> <td>15</td> </tr> <tr> <td>01</td> <td>44</td> <td>20</td> </tr> <tr> <td>02</td> <td>22</td> <td>5</td> </tr> <tr> <td>02</td> <td>44</td> <td>10</td> </tr> </tbody> </table>	SOC	DEP	Count(*)	01	22	15	01	44	20	02	22	5	02	44	10									
SOC	DEP	Count(*)																								
01	22	15																								
01	44	20																								
02	22	5																								
02	44	10																								
<p>GROUPING SETS</p>	<p>affiche les totaux pour 2 GROUPES consécutifs</p>	<p>SELECT SOC, DEP, Count(*) ...GROUP BY <b>Grouping Sets</b> (soc, Dep)</p> <table border="1" data-bbox="927 754 1129 1120"> <thead> <tr> <th>SOC</th> <th>DEP</th> <th>Count(*)</th> </tr> </thead> <tbody> <tr> <td>01</td> <td>-</td> <td>35</td> </tr> <tr> <td>02</td> <td>-</td> <td>15</td> </tr> <tr> <td>-</td> <td>22</td> <td>20</td> </tr> <tr> <td>-</td> <td>44</td> <td>30</td> </tr> </tbody> </table>	SOC	DEP	Count(*)	01	-	35	02	-	15	-	22	20	-	44	30									
SOC	DEP	Count(*)																								
01	-	35																								
02	-	15																								
-	22	20																								
-	44	30																								
<p>ROLLUP</p>	<p>affiche 1 total par groupe puis des ruptures de niveau supérieur</p>	<p>SELECT SOC, DEP, Count(*) ... GROUP BY <b>ROLLUP</b> (soc, Dep)</p> <table border="1" data-bbox="927 1245 1158 1821"> <thead> <tr> <th>SOC</th> <th>DEP</th> <th>Count(*)</th> </tr> </thead> <tbody> <tr> <td>01</td> <td>22</td> <td>15</td> </tr> <tr> <td>01</td> <td>44</td> <td>20</td> </tr> <tr> <td>01</td> <td>-</td> <td>35</td> </tr> <tr> <td>02</td> <td>22</td> <td>5</td> </tr> <tr> <td>02</td> <td>44</td> <td>10</td> </tr> <tr> <td>02</td> <td>-</td> <td>15</td> </tr> <tr> <td>-</td> <td>-</td> <td>50</td> </tr> </tbody> </table>	SOC	DEP	Count(*)	01	22	15	01	44	20	01	-	35	02	22	5	02	44	10	02	-	15	-	-	50
SOC	DEP	Count(*)																								
01	22	15																								
01	44	20																								
01	-	35																								
02	22	5																								
02	44	10																								
02	-	15																								
-	-	50																								
<p>CUBE</p>	<p>affiche les totaux pour tous les groupes possibles</p>	<p>SELECT SOC, DEP, Count(*) GROUP BY <b>CUBE</b> (soc, Dep)</p> <table border="1" data-bbox="927 1951 1270 2092"> <thead> <tr> <th>SOC</th> <th>DEP</th> <th>Count(*)</th> </tr> </thead> <tbody> <tr> <td>01</td> <td>22</td> <td>15</td> </tr> </tbody> </table>	SOC	DEP	Count(*)	01	22	15																		
SOC	DEP	Count(*)																								
01	22	15																								

		<table border="1"> <tr><td>01</td><td>44</td><td>20</td></tr> <tr><td>01</td><td>-</td><td>35</td></tr> <tr><td>02</td><td>22</td><td>5</td></tr> <tr><td>02</td><td>44</td><td>10</td></tr> <tr><td>02</td><td>-</td><td>15</td></tr> <tr><td>-</td><td>-</td><td>50</td></tr> <tr><td>-</td><td>22</td><td>20</td></tr> <tr><td>-</td><td>44</td><td>30</td></tr> </table>	01	44	20	01	-	35	02	22	5	02	44	10	02	-	15	-	-	50	-	22	20	-	44	30								
01	44	20																																
01	-	35																																
02	22	5																																
02	44	10																																
02	-	15																																
-	-	50																																
-	22	20																																
-	44	30																																
GROUPING()	indique si cette ligne est le résultat de ROLLUP ( <i>rupture</i> )	<p>SELECT SOC, DEP, Count(*), GROUPING(DEP) GROUP BY <b>ROLLUP</b> (soc, Dep)</p> <table border="1"> <thead> <tr><th>SOC</th><th>DEP</th><th>Count(*)</th><th>Grouping(dep)</th></tr> </thead> <tbody> <tr><td>01</td><td>22</td><td>15</td><td>0</td></tr> <tr><td>01</td><td>44</td><td>20</td><td>0</td></tr> <tr><td>01</td><td>-</td><td>35</td><td>1</td></tr> <tr><td>02</td><td>22</td><td>5</td><td>0</td></tr> <tr><td>02</td><td>44</td><td>10</td><td>0</td></tr> <tr><td>02</td><td>-</td><td>15</td><td>1</td></tr> <tr><td>-</td><td>-</td><td>50</td><td>1</td></tr> </tbody> </table>	SOC	DEP	Count(*)	Grouping(dep)	01	22	15	0	01	44	20	0	01	-	35	1	02	22	5	0	02	44	10	0	02	-	15	1	-	-	50	1
SOC	DEP	Count(*)	Grouping(dep)																															
01	22	15	0																															
01	44	20	0																															
01	-	35	1																															
02	22	5	0																															
02	44	10	0																															
02	-	15	1																															
-	-	50	1																															
<u>7.3 : options d'agrégation*</u>	les fonctions d'agrégation peuvent être utilisées comme des fonctions OLAP (avec OVER) et sans GROUP BY	<p>SELECT codart, prix, sum(prix) over(order by codart) from articles</p> <table border="1"> <thead> <tr><th>CODART</th><th>Prix</th><th>SUM(prix)</th></tr> </thead> <tbody> <tr><td>01</td><td>4</td><td>4</td></tr> <tr><td>02</td><td>15</td><td>19</td></tr> <tr><td>03</td><td>11</td><td>30</td></tr> <tr><td>04</td><td>7</td><td>37</td></tr> <tr><td>...</td><td>...</td><td>...</td></tr> </tbody> </table>	CODART	Prix	SUM(prix)	01	4	4	02	15	19	03	11	30	04	7	37	...	...	...														
CODART	Prix	SUM(prix)																																
01	4	4																																
02	15	19																																
03	11	30																																
04	7	37																																
...	...	...																																



## Fonctions XML (V7)

XMLDOCUMENT()	production d'un flux XML à partir d'une chaîne de caractère. Cette action est implicite lors des ordres INSERT et UPDATE, dans une colonne de type XML.	
XMLPARSE()	production après vérification, d'un flux XML, avec choix de conservation des espaces ou non	XMLPARSE(DOCUMENT '<xml> ... </xml>') PRESERVE WHITESPACE
XMLVALIDATE()	validation d'un flux XML à l'aide d'un schéma XSD enregistré dans XSROBJECTS	XMLVALIDATE(DOCUMENT '<xml> ... </xml>' ACCORDING TO XMLSCHEMA <schema>)
XSLTRANSFORM( )	transforme un flux XML à l'aide de XSLT	XSLTRANSFORM( flux-xml USING 'source-XSLT')
XMLTEXT( )	retourne un texte compatible XML	select XMLTEXT('100 est > à 99 & à 98') FROM SYSIBM.SYSDUMMY1 ; ==> 100 est &gt; à 99 & à 98
XMLELEMENT( )	production d'un élément XML	select XMLELEMENT(name "numero" , nocli) from clients  <numero>1</numero> <numero>2</numero>
XMLNAMESPACE( )	génération d'un espace de nommage	select xmlelement(name "client:nom", XMLNAMESPACES('http://www.volubis.fr/clients/1.0' AS "client") , raisoc) from clients;  <client:nom xmlns:client="http://www.volubis.fr/clients/1.0">IBM</client:nom> <client:nom xmlns:client="http://www.volubis.fr/clients/1.0">Rational</client:nom>
XMLPI( )	balise <i>processing instruction</i>	SELECT XMLPI(NAME "Instruction", 'APPUYEZ SUR ENTREE') FROM SYSIBM.SYSDUMMY1 <?Instruction APPUYEZ SUR ENTREE?>

XMLCOMMENT,	commentaire XML	<pre>select XMLCOMMENT('A consommer avec modération') FROM SYSIBM.SYSDUMMY1 ; &lt;!--A consommer avec modération--&gt;</pre>
XMLCONCAT( )	Concatenation de deux flux XML	<pre>select XMLCONCAT( XMLELEMENT(name "numero", nocli) , XMLELEMENT(name "nom", raisoc) ) from clients  &lt;numero&gt;1&lt;/numero&gt;&lt;nom&gt;IBM&lt;/nom&gt; &lt;numero&gt;2&lt;/numero&gt;&lt;nom&gt;Rational&lt;/nom&gt;</pre>
XMLFOREST( )	Suite d'éléments XML à partir des colonnes d'une table	<pre>select XMLFOREST(nocli , raisoc) from clients  &lt;NOCLI&gt;1&lt;/NOCLI&gt;&lt;RAISOC&gt;IBM&lt;/RAISOC&gt; &lt;NOCLI&gt;2&lt;/NOCLI&gt;&lt;RAISOC&gt;Rational&lt;/RAISOC&gt;</pre>
XMLROW( )	arborescence XML à partir des colonnes d'une table	<pre>select XMLROW(nocli , raisoc) from clients  &lt;row&gt; &lt;NOCLI&gt;1&lt;/NOCLI&gt;&lt;RAISOC&gt;IBM&lt;/RAISOC&gt; &lt;/row&gt; &lt;row&gt; &lt;NOCLI&gt;2&lt;/NOCLI&gt;&lt;RAISOC&gt;Rational&lt;/RAISOC&gt; &lt;/row&gt;</pre>
XMLAGG( )	Fonction d'agrégation, éléments XML par groupe (GROUP BY) ou pour la totalité du fichier (sans )	<pre>select xmlagg(XMLELEMENT(name "nom" , raisoc)) from clients  &lt;nom&gt;IBM&lt;/nom&gt;&lt;nom&gt;Rational&lt;/nom&gt; ....</pre>
XMLGROUP( )	Fonction d'agrégation, arborescences XML par groupe (GROUP BY) ou pour la totalité du fichier (sans GROUP BY)	<pre>select xmlgroup(XMLELEMENT(name "nom" , raisoc)) from clients  &lt;rowset&gt; &lt;row&gt;&lt;NOCLI&gt;1&lt;/NOCLI&gt;&lt;RAISOC&gt;IBM&lt;/RAISOC&gt;&lt;VILLE&gt;New York &lt;/VILLE&gt;&lt;/row&gt; &lt;row&gt;&lt;NOCLI&gt;2&lt;/NOCLI&gt;&lt;RAISOC&gt;Rational&lt;/RAISOC&gt;&lt;VILLE&gt;Toronto &lt;/VILLE&gt;&lt;/row&gt; &lt;/rowset&gt;</pre>
XMLTABLE( )	<p>Fonction Table qui traite sous forme colonnée (relationnelle), les éléments d'un flux XML. La source peut être :</p> <ul style="list-style-type: none"> <li>• une colonne de type XML</li> <li>• le résultat de GET_XML_FILE</li> <li>• le résultat de HTTPGETBLOB CLOB</li> </ul>	<pre>SELECT X.NOM, X.RUE, X.TEL FROM XMLTABLE ('\$c/customerinfo' passing &lt;source&gt; as "c" COLUMNS NOM CHAR(30) PATH 'name', RUE VARCHAR(25) PATH 'addr/street', TEL VARCHAR(20) PATH '@tel' ) AS X</pre>

Vous pouvez aussi utiliser les nouvelles fonctions SQL pour insérer du XML dans une table :

GET_BLOB_FROM_FILE(chemin , option)	retourne un BLOB LOCATOR, sans conversion du CCSID
GET_CLOB_FROM_FILE(chemin , option)	retourne un CLOB LOCATOR dans le CCSID du job
GET_DBCLOB_FROM_FILE(chemin , option)	retourne un DBCLOB LOCATOR dans le CCSID DBCS par défaut
GET_XML_FILE(chemin)	retourne un BLOB LOCATOR en UTF-8, si ce dernier ne possède pas de déclaration XML la fonction l'ajoute.
-> s'utilisent aussi en programmation	<pre>EXEC SQL   values cast(GET_CLOB_FROM_FILE('/monfichier.txt')             as varchar(20000) ) into :variable;</pre>

Les deux fonctions suivantes sont disponibles, si vous avez installé Omnifind (5733OMF) et créé un Index (*CALL SYPROCS.SYSTS\_CREATE*)

CONTAINS(zone, 'recherche')	retourne 1 si la recherche est présente dans zone
SCORE(zone, 'recherche')	retourne le score (degré de pertinence, entre 0 et 1)

Ces deux fonctions peuvent avoir un troisième argument **options**, construit comme suit :

- QUERYLANGUAGE= fr\_FR ou en\_US
- RESULTLIMIT=n, pour limiter la réponse aux **n** premières valeurs.
- SYNONYM = OFF ou ON, pour utiliser ou non les synonymes.

## Fonctions JSON (7.2/7.3)

JSON_TABLE()	<p>Fonction Table qui traite sous forme colonnée (relationnelle), les éléments d'un flux JSON. La source peut être :</p> <ul style="list-style-type: none"> <li>• une colonne de type VARCHAR</li> <li>• le résultat de GET_CLOB_FROM_FILE</li> <li>• le résultat de HTTPGETCLOB</li> </ul>	<pre>SELECT X.NOM, X.RUE, X.TEL FROM JSON_TABLE (&lt;source&gt; , \$.client[*] COLUMNS ( NOM CHAR(30) PATH 'lax \$.name', RUE VARCHAR(25) PATH 'lax \$.addr.street', TEL VARCHAR(20) PATH 'strict \$.tel') ) AS X</pre>
JSON_VALUE()*	retourne une valeur scalaire (unitaire)	<pre>VALUES (JSON_VALUE( Json_VAR, '\$.id' RETURNING Integer) ==&gt; 901;</pre>
JSON_QUERY()*	retourne une chaîne au format JSON	<pre>VALUES (JSON_QUERY( Json_VAR, '\$.name') =&gt; {"first":"John","last":"Doe"};</pre>
JSON_ARRAY()*	<i>Produit un tableau de valeur (rappel entre [ et ])</i>	<pre>VALUES (JSON_ARRAY ( (SELECT DEPTNO FROM DEPT WHERE DEPTNAME LIKE 'BRANCH OFFICE%'))); ==&gt; ["F22","G22","H22","I22","J22"]</pre>
JSON_OBJECT()*	<i>produit un objet JSON, chaque élément a un nom et une valeur</i>	<pre>SELECT JSON_OBJECT('Nom' : LASTNAME, 'date naissance' : HIREDATE, 'Salaire' SALARY) FROM EMPLOYEE WHERE EMPNO = '000020' ==&gt;{"Nom":"THOMPSON","date naissance":"1973-10- 10","Salaire":41250.00}</pre>
JSON_ARRAYAGG()*	<i>produit un tableau de valeurs par groupe (GROUP BY)</i>	<pre>SELECT workdept, JSON_ARRAYAGG(lastname) FROM EMPLOYEE WHERE workdept LIKE 'D%' GROUP BY workdept;</pre>
JSON_OBJECTAGG()*	<i>produit une série d'objets JSON (clé/valeur) par groupe (GROUP BY)</i>	<pre>SELECT JSON_OBJECTAGG(workdept, JSON_OBJECTAGG(char(empno) value lastname) FROM EMPLOYEE WHERE workdept LIKE 'D%' GROUP BY workdept;</pre>

## Fonctions HTTP de Systools

HTTPHEAD(url, httpheader ou chaîne vide [''])	<p>passé une requête HTTP retourne l'entête HTTP (header) retournée par le serveur</p>
HTTPBLOB(url, GET   POST   PUT   DELETE, httpheader, data )	<p>passé une requête HTTP avec la méthode indiquée et retourne la réponse sous forme de BLOB</p>
HTTPCLOB(url, GET   POST   PUT   DELETE, httpheader, data )	<p>passé une requête HTTP avec la méthode indiquée et retourne la réponse sous forme de CLOB</p>
HTTPGETBLOB(url, httpheader ou chaîne vide [''])	<p>passé une requête HTTP avec la méthode GET et retourne la réponse sous forme de BLOB</p>
HTTPGETCLOB(url, httpheader ou chaîne vide [''])	<p>passé une requête HTTP avec la méthode GET et retourne la réponse sous forme de CLOB</p>
HTTPPOSTBLOB(url, httpheader ou chaîne vide [''])	<p>passé une requête HTTP avec la méthode POST et retourne la réponse sous forme de BLOB</p>
HTTPPOSTCLOB(url, httpheader ou chaîne vide [''])	<p>passé une requête HTTP avec la méthode POST et retourne la réponse sous forme de CLOB</p>
HTTPPUTBLOB(url, httpheader ou chaîne vide [''])	<p>passé une requête HTTP avec la méthode PUT et retourne la réponse sous forme de BLOB</p>
HTTPPUTCLOB(url, httpheader ou chaîne vide [''])	<p>passé une requête HTTP avec la méthode PUT et retourne la réponse sous forme de CLOB</p>
HTTPDELETEBLOB(url, httpheader ou chaîne vide [''])	<p>passé une requête HTTP avec la méthode DELETE et retourne la réponse sous forme de BLOB</p>
HTTPDELETECLOB(url, httpheader ou chaîne vide [''])	<p>passé une requête HTTP avec la méthode DELETE et retourne la réponse sous forme de CLOB</p>
URLENCODE(chaîne, encodage [UTF-8 par défaut])	<p>retourne la chaîne au format compatible avec une URL</p> <pre>systools.urlencode('info@volubis.fr', '') - &gt; info%40volubis.fr</pre>

URLDECODE(chaine, encodage [UTF-8 par défaut])	retourne la chaîne en clair
BASE64ENCODE(chaine)	retourne la chaîne au format base64 (utilisé dans les entêtes HTTP pour l'authentification)
BASE64DECODE(chaine)	retourne la chaîne en clair
<pre>--exemple, retourne la liste des taux de change avec l'EURO depuis la BCE SELECT * FROM XMLTABLE('\$result/*:Envelope/*:Cube/*:Cube/*:Cube' PASSING XMLPARSE( DOCUMENT  SYSTOOLS.HTTPGETCLOB('http://www.ecb.europa.eu/stats/eurofxref/eurofxref-daily.xml','') ) as "result" COLUMNS monnaie CHAR(3) PATH '@currency', taux DEC(11, 4) PATH '@rate' ) AS LESTAUX;</pre>	
<pre>--Toutes ces fonctions sont aussi livrées en mode "verbeux" (fonctions TABLE retournant le contenu <b>et</b> l'entête HTTP)  select * from TABLE ( systools.httpgetclobVERBOSE('http://www.volubis.fr' , '') ) as T +-----+-----+ + RESPONSEMSG + RESPONSEHTTPHEADER + + &lt;!DOCTYPE html PUBLIC..+ &lt;httpheader responseCode="200" + +-----+-----+</pre>	

## Expressions [régulières](#) (version 7.2)

REGEXP_COUNT()	Compte le nombre de fois ou une expression régulière est vraie	Where REGEXP_COUNT(pr_nom , 'ch[aâ]teau') > 1 // présent au moins 2 fois
REGEXP_INSTR()	retourne la position de la chaîne où l'expression régulière est vraie	Where REGEXP_INSTR(pr_nom , 'ch[aâ]teau') > 5 // Chateau après position 5
REGEXP_SUBSTR()	retourne la chaîne qui fait que l'expression régulière est vraie	VALUES REGEXP_SUBSTR(msg , '(\w+\.)+((org) (com) (gouv) (fr))') into :machaine // une URL valide
REGEXP_REPLACE()	remplace la chaîne qui fait que l'expression régulière est vraie	REGEXP_REPLACE(pr_nom , 'ch[aâ]teau' , 'bodega')

## Cas particulier des dates

- On ne peut utiliser l'arithmétique temporelle qu'avec des dates, des heures, des horodatages
- les calculs peuvent se faire sous la forme
  1. date + durée = date
  2. date - durée = date
  3. date - date = durée
  4. heure + durée = heure
  5. etc ..
- les durées peuvent être exprimées de manière explicite avec
  -

YEARS	MONTHS	DAYS
HOURS	MINUTES	SECONDS

- les durées résultat (DATLIV - DATCDE) seront toujours exprimées sous la forme AAAAMMJJ, où :
  -

AAAA	représente le nombre d'années
MM	le nombre de mois
JJ	le nombre de jours

- 
- Ainsi, si SQL affiche **812**, il faut comprendre **8 mois, 12 jours**
- 40301 signifie **4** ans , **03** mois, **01** jour (attention SQL risque d'afficher 40.301)

## Fonctions liées aux dates

<i>Fonction(x)</i>	<i>Retourne ?</i>	<i>Exemple</i>
DATE(x) X doit être une chaîne au format SQL (ISO fonctionne toujours)	une date ( <i>sur laquelle les fonctions suivantes s'appliquent</i> )	DATE ( substr (digits (dat8) ,1,4) concat '-' concat substr (digits (dat8) ,5,2) concat '-' concat substr (digits (dat8) ,7,2) )
DAY(D) DAYOFMONTH(D)	retourne la partie jour de D (doit être une date ou un écart AAAAMMJJ).	DAY (DATCDE)
MONTH(D)	retourne la partie mois de D (idem)	MONTH (current date)
YEAR(D)	Retourne la partie année de D (idem)	YEAR (current date - DATCDE)
DAYOFYEAR(D)	retourne le n° de jour dans l'année (julien)	DAYOFYEAR (datdep)
DAYOFWEEK(D)	retourne le N° de jour dans la semaine (1 = Dimanche, 2=Lundi, ...)	DAYOFWEEK (ENTRELE)
DAYOFWEEK_ISO(D)	retourne le N° de jour dans la semaine (1 = Lundi, ...)	DAYOFWEEK_ISO (ENTRELE)
DAYNAME(d)	retourne le nom du jour de d (Lundi, Mardi, ...)	DAYNAME (datcde)
MONTHNAME(d)	retourne le nom du mois de d (Janvier, Février, ...)	MONTHNAME (datcde)
EXTRACT(day from d)	Extrait la partie jour de D (aussi MONTH et YEAR)	EXTRACT (MONTH from datcde)
DAYS(D)	retourne le nbr de jours depuis 01/01/0001	DAYS (datcde) - DAYS (datliv)
QUARTER(D)	retourne le n° du trimestre	QUARTER (DATEFIN)
WEEK(D)	retourne le n° de semaine ( <i>Attention 01/01/xx donne toujours semaine 1</i> )	WHERE WEEK (DATLIV) = WEEK (DATCDE)



WEEK_ISO(D)	retourne le n° de semaine (la semaine 1 est celle qui possède un JEUDI dans l'année.)	WHERE WEEK_ISO (DATLIV) = WEEK_ISO (DATCDE)
CURDATE()	retourne la date en cours, comme CURRENT DATE	
CURTIME()	retourne l'heure en cours, comme CURRENT TIME	
NOW()	retourne le timestamp en cours	
JULIAN_DAY(d)	retourne le nbr de jours qui sépare une date du <a href="#">1er Janv. 4712</a> avant JC.	JULIAN_DAY (datcde)
LAST_DAY(d)	retourne la date correspondant au dernier jour du mois.	LAST_DAY ('2006-04-21') = 2006-04-30
ADD_MONTHS(d, nbr )	ajoute un nbr de mois à une date , si la date est au dernier jour du mois, la date calculée est aussi au dernier jour du mois	ADD_MONTHS ('2006-04-30' , 1) = 2006-05-31
NEXT_DAY(d, 'day' )	retourne la prochaine date ayant le jour demandé	NEXT_DAY ('2006-12-31' , 'DIM') = ' 2007-01-07'
MONTHS_BETWEEN(d, d)	retourne l'écart en mois (avec des décimales sur 31 jours) entre deux dates	months_between ('25/09/08' , '31/08/08') = 0,806451612903225

## Fonctions liées aux heures

Fonction(x)	Retourne ?	Exemple
TIME(T)	une heure	TIME ( substr (digits (h6) ,1,2) concat ':' concat substr (digits (h6) ,3,2) concat ':' concat substr (digits (h6) ,5,2) )
HOUR(T)	retourne la partie heure de T	HOUR (Pointage)
MINUTE(T)	retourne la partie minute de T	
SECOND(T)	Retourne la partie secondes de T	

EXTRACT(hour from t)	la partie heure de T (aussi MINUTE et SECOND)	EXTRACT(SECOND from pointage)
----------------------	---	-------------------------------

## Fonctions liées aux Timestamp

Fonction(x)	Retourne ?	Exemple										
TIMESTAMP(T)	un timestamp (date - heure - microsecondes)	TIMESTAMP ( ' 1999-10-06-15.45.00.000001 ' )										
TIMESTAMP (D T)	un timestamp (microsecondes à 0)	TIMESTAMP (datecde heure)										
TIMESTAMP_ISO(x)	un timestamp à partir de x Si x est une date, l'heure est à 00:00:00 Si x est une heure, la date est à aujourd'hui.	TIMESTAMP_ISO (heure_pointage)										
TIMESTAMPDIFF (c 'DIFFERENCE')	C indique l'unité de mesure de l'écart que vous souhaitez obtenir <table border="1" data-bbox="550 1048 877 1541"> <tr> <td>1 = fractions de s.</td> <td>16 = jours</td> </tr> <tr> <td>2 = secondes</td> <td>32 = semaines</td> </tr> <tr> <td>4 = minutes</td> <td>64 = mois</td> </tr> <tr> <td>8 = heures</td> <td>128 = trimestres</td> </tr> <tr> <td></td> <td>256 = Année</td> </tr> </table> 'DIFFERENCE' est la représentation caractères [char(22) ] d'un écart entre deux timestamp.	1 = fractions de s.	16 = jours	2 = secondes	32 = semaines	4 = minutes	64 = mois	8 = heures	128 = trimestres		256 = Année	TIMLESTAMPDIFF (32 , CAST (CURRENT_TIMESTAMP - CAST (DATLIV AS TIMESTAMP) AS CHAR (22)) )  indique l'écart en semaines entre DATLIV et aujourd'hui
1 = fractions de s.	16 = jours											
2 = secondes	32 = semaines											
4 = minutes	64 = mois											
8 = heures	128 = trimestres											
	256 = Année											
MIDNIGHT_SECONDS	retourne le nbr de secondes qui sépare un timestamp de minuit	MIDNIGHT_SECONDS (pointage)										
VARCHAR_FORMAT(d, 'YYYY-MM-DD HH24:MI:SS' )	Transforme un timestamp en chaine (le format est imposé en V5R40, libre en V6)	VARCHAR_FORMAT ( now() , 'YYYY-MM-DD HH24:MI:SS' )										

<p>TIMESTAMP_FORMAT('c', f)</p>	<p>Transforme une chaîne <i>c</i> en timestamp suivant le format <i>f</i></p> <p><b><i>f</i> pouvant contenir :</b></p> <ul style="list-style-type: none"> <li>• - . / , ' : ; et (espace)</li> <li>• DD (jours) MM (mois) YY (années sur 2) YYYY (sur 4)</li> <li>• RR année ajustée (00 à 49&gt;2000, 50 à 99&gt;1900)</li> <li>• HH24 l'heure (24h)</li> <li>• SS (secondes)</li> <li>• NNNNNN (micro-secondes)</li> </ul>	<pre>TIMESTAMP_FORMAT('99/02/05' , 'RR/MM/DD') =1999-02-05-00.00.00.000000</pre> <p><i>le format YY/MM/DD aurait donné 2099</i></p>
<p>GENERATE_UNIQUE()</p>	<p>génère une valeur unique de type CHAR(13) basée sur le timestamp en cours.</p>	<pre>insert GENERATE_UNIQUE() ....</pre>
	<p><b><i>plus toutes les fonctions liées aux dates et aux heures</i></b></p>	

**Sous sélections**, Ordre SQL intégré dans la clause WHERE (ou dans la liste des colonnes) d'un ordre SQL :

```
SELECT * FROM tarif WHERE prix < (SELECT AVG(prix) from tarif)
```

*Donne la liste des articles ayant un prix inférieur à la moyenne*

---

```
SELECT * FROM tarif WHERE prix BETWEEN  
(SELECT AVG(prix)*0,9 from tarif) AND (SELECT  
AVG(prix)*1,1 from tarif)
```

*donne la liste des articles ayant un prix variant d'au maximum +/- 10 % par rapport à la moyenne*

---

```
SELECT * FROM tarif T WHERE prix <  
(SELECT AVG(prix) from tarif Where famille = t.famille)
```

*Donne la liste des articles ayant un prix inférieur à la moyenne de leur famille*

---

```
Select codart , (qte * prix) as montant,  
      (select sum(qte * prix) from commandes where  
      famcod = c1.famcod) as global_famille  
from commandes c1
```

*Donne la liste des commandes (article, montant commandé), en rappelant sur chaque ligne le montant **global** commandé dans la famille.*

---

- vous pouvez aussi utiliser la clause **EXISTS** dans un SELECT imbriqué.
- Elle indique VRAI si le select imbriqué retourne une ligne (ou plus)
- Elle indique FAUX si le select imbriqué ne retourne aucune ligne.

*Soit un fichier article ayant une colonne STOCKAGE (O/N) et un fichier stock, si vous voulez supprimer les articles dans le fichiers stock ayant la zone **stockage** à 'N' dans le fichier **article**.*

```
DELETE from stock S where exists  
(SELECT * from articles where codart = S.codart and stockage = 'N')
```