AF 400
AUTOFORMATION ONLINE SUR IBMI ET POWER SYSTEM

Mémo SQL

* => nouveauté 7.4

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 join fichier2 f2 ON f1.clea = f2.clea
                                  and f1.cleb = f2.cleb
                [join fichier3 f3 on f2.clec = f3.clec]
```

options de jointure

Join	*uniquement les enregistrements en correspondance*
Left outer join	*tous les enregistrements de fichier1*
exception join	*uniquement les enregistrements sans correspondance*
right outer join	*tous les enregistrements de fichier2*
right exception join	*uniquement les enregistrements sans correspondance*
full outer join	*toutes les combinaisons (right et left outer)*

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

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

options de limitation

```
     FETCH FIRST n ROWS ONLY (affiche les n premières lignes)
ou
     LIMIT n OFFSET Y (affiche les n premières lignes à partir de y+1)
```

(tables dérivées)

Soit

```
SELECT colonne1 [as entete1],colonne2 [as entete2]
FROM (SELECT ... FROM ...) as nom-temporaire WHERE [sélection]
```

Soit

```
WITH nom-temporaire as (SELECT ... FROM ...)
  SELECT colonne1 [as entete1],colonne2 [as entete2]
   FROM nom-temporaire WHERE [sélection]
```

(Requête récursive)

```
 WITH temp (chef, matricule, nom)
 as (select chef, matricule, nom
      from personnel  where chef is null
 UNION ALL
     select P.chef, P.matricule, P.nom
      from temp T join personnel P
      on T.matricule = P.chef
    )
SELECT * FROM TEMP
```

(Requête hiérarchique)

```
SELECT chef, matricule, nom
       FROM personnel
 START WITH chef is NULL
 CONNECT BY PRIOR matricule = chef
```

options liées :

ORDER SIBLINGS BY nom	Tri les lignes ayant le même parent
CONNECT BY **NOCYCLE** PRIOR	évite l'arrêt en erreur lors d'une boucle infinie (la ligne qui provoque la boucle est affichée une deuxième fois)
CONNECT_BY_ISCYCLE	retourne 1 si la ligne en cours aurait provoqué une boucle
CONNECT_BY_ISLEAF	retourne 1 si la ligne en cours n'a pas d'enfant
LEVEL	indique le niveau dans la hiérarchie pour cette ligne
CONNECT_BY_ROOT	indique l'élément racine (le parent d'origine) pour cette ligne
SYS_CONNECT_BY_PATH (<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 CLOB : /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
 Select * From fichier
     for system_time as of '2016-02-10-12.00.00.00000000000'
```

ou

```
 Select * From fichier
      from '2016-02-01-00.00.00.00000000000'
        to '2016-02-10-23.59.59.00000000000'
```

UPDATE *(-> V4R20)*	**UPDATE** fichier **SET** colonne1 = valeur1 **WHERE** [sélection]
UPDATE *(V4R30 et +)*	**UPDATE** fichier f **SET** colonne1 = (select valeur1 from autrefichier where cle = f.cle) **WHERE** [sélection] **Order** BY x **LIMIT** n **OFFSET** y
INSERT	**INSERT INTO** fichier **VALUES**(valeur1, *touteslescolonnes*...) *ou* **INSERT INTO** fichier (colonne2, colonne3) **VALUES**(v2, v3) *ou* **INSERT INTO** fichier (**SELECT ... FROM ...WHERE ...**)
INSERT + FINAL TABLE	**SELECT cle_auto, quand FROM FINAL TABLE (INSERT INTO fournisseur** **(raisoc, quand) VALUES('IBM', now()))**-- *affiche la nouvelle ligne insérée*
DELETE	**DELETE FROM** fichier **WHERE** [sélection] **Order** BY x **LIMIT** n **OFFSET** y *
TRUNCATE (*CLRPFM*)	**TRUNCATE** TABLE CLIENTP1 **DROP STORAGE** *IGNORE DELETE TRIGGERS RESTART IDENTITY*
VALUES	**VALUES now() -- *affiche le timestamp en cours***
MERGE	**MERGE INTO cible C** **USING (**SELECT source1 , source2 FROM source**) S** **ON (C.zonecle = S.zonecle)** **WHEN MATCHED THEN** UPDATE SET cible2 = source2 **WHEN NOT MATCHED THEN** INSERT (cible1, cible2) VALUES(source1, source2)
Concurrence d'accès pour SELECT, UPDATE, DELETE et MERGE	**Suivant le niveau de commitment control** **- WAIT FOR OUTCOME** Attendre que les lignes verrouillées soient libérées (CS, ou RS) **- SKIP LOCKED DATA** les lignes verrouillées sont ignorées (NC, UR, CS, ou RS) **- USE CURRENTLY COMMITTED** Utiliser les (anciennes) valeurs déjà validées (SELECT sous CS uniquement)

Sélections

Opérateur logique	Exemple(s)

colonne ***op***. colonne
ou
colonne ***op***. valeur
 	
op.
=
<
>
<>
<=
>=
	`QTECDE <> 0`
	
	`LIBART = 'Pamplemousse'`
	
	`PRIX >= 45`

les valeurs peuvent être comparées « en ligne »	... where (cepage1, cepage2) = ('Syrah' , 'Grenache')
IN (val1, val2, val3, ...)	`DEPT IN (44, 49, 22, 56, 29)`
BETWEEN val1 **AND** val2	`DEPT BETWEEN 44 AND 85`

LIKE	`nom LIKE 'DU%'  (commence par)`
	`nom LIKE '%PON%' (contient)`
	`nom LIKE '%RAND' (se termine par)`
	
	aussi : `nom LIKE '%'concat` *ville* `concat '%'`

IS NULL | ISNULL * **IS NOT NULL | NOTNULL** *	`test la valeur nulle (`*pratique avec les jointures externes*`)`
et aussi **OR**, **AND**, **NOT**, **(**, **)**.	`CODART = 1245 or LIBART = 'Pamplemousse'`
REGEXP_LIKE(zone, <expression régulière>)	`Where REGEXP_LIKE(pr_nom , 'ch[aâä]teau')`
IS (NOT) JSON	`Where JDATA IS JSON`
JSON_EXISTS	`Where JSON_EXISTS(JDATA` ` 'strict $.TEL'  FALSE ON ERROR)`
EXISTS	`Where EXISTS (select * from …)` *– voir la fin de ce mémo*

Fonctions scalaires *(ligne à ligne)*

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() | RANDOM()*	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 chaine une information numérique	`VALUES VARCHAR_FORMAT(1,5 - 1,4, '0D0')` `-> '0,1'`

	avec dans le masque : 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	
BLOB(x)	une chaîne de car. (x) en tant que BLOB	`BLOB('texte de la chaine à convertir')`
CLOB(x) |TO_CLOB(x) *	une chaîne de car. (x) en tant que CLOB	`CLOB('texte de la chaine 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])	x au format indiqué par typeSQL :
	
	types valides :
	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
	
	** (FR = 297, US = 37)*
		`CAST(qte AS CHAR(9))`
		Attention les zéros de gauche sont éliminés
		
		`CAST(prixchar as NUMERIC(7, 2))`
		
		`cast('123456,89' as numeric(8, 2))`
		fonctionne
		
		`cast('123456,89' as numeric(7, 2))`
		donne une erreur (trop d'entiers)

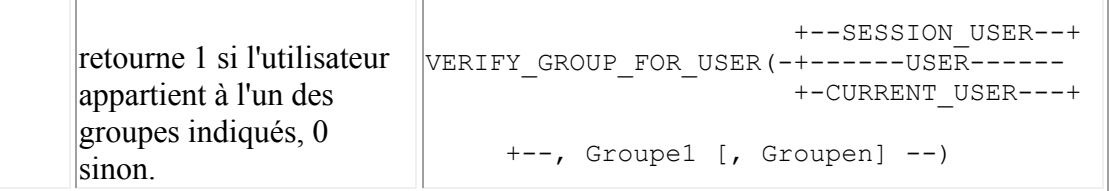
INTERPRET(x AS typeSQL[lg]) *	force SQL à "caster" une chaîne, selon le type indiqué	`Select interpret(`
		` substr(entry_data , 1 , 4) as integer)`
		`from table(QSYS2.display_journal( ))`

STRIP | TRIM(x) RTRIM(x [, 'c']) LTRIM(x [, 'c'])	Supprime les blancs d'extrémité blancs (ou c) de droite blancs (ou c) de gauche	`TRIM(raisoc)`

LENGTH(x) ou
OCTET_LENGTH(x)
	la longueur de x	`LENGTH(nom)`
		
		`LENGTH(TRIM(nom))`

CONCAT(x , y)	concatene X et Y (*aussi* x CONCAT y *ou* X || Y)	`CONCAT(nom, prenom)`
SUBSTR(x, d, l)	extrait une partie de x depuis D sur L octets	`SUBSTR(nom, 1, 10)` `SUBSTR(nom, length(nom), 1)`
LEFT(x, l) | STRLEFT(x, l) *	extrait une partie de x depuis 1 sur L octets	`LEFT(nom, 10)`
RIGHT(x, l) | STRRIGHT(x, l) *	extrait les L derniers octets de x	`RIGHT(nom, 5)`
SPACE(n)	retourne n blancs	`nom concat space(5) concat prenom`
REPEAT(x , n)	retourne n fois x	`repeat('*', 15)`
RPAD(chaine , n , 'c')	Complète une chaîne à droite par le caractère indiqué (c) jusqu'à longueur de n	`RPAD(PR_TEL, 15, '.')`
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)`
HASH_ROW(fichier) *	retourne la valeur du codage SHA512 d'une ligne	`select rrn(C), HASH_ROW(C) from clientp1 C`
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 ***origine*** par le caractère de même position dans ***remplace.***	`TRANSLATE(prixc, ' €', '0$')` *`remplace 0 par espace et $ par €`*
REPLACE(x, origine, remplacement)	Remplace la chaîne de caractère ***origine*** par la chaîne ***remplacement*** *dans x*	`REPLACE(x, 'Francs' , 'Euros')` *`--remplace`* ***`Francs`*** *`par`* ***`Euros`***
REPLACE(x, origine)	Supprime la chaîne de caractère ***origine***	`REPLACE(x, 'Francs' )` *`--supprime`* ***`Francs`***
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` *`--0 = très différents, 4 = trés proches`*
COALESCE(x,y,[z]) | NVL(x,y,[z]) *	retourne première valeur non nulle	`COALESCE(DEPCLI, DEPLIV, 0)`
IFNULL(x, y) | VALUE(x, y)	retourne X s'il est non null, sinon Y	`IFNULL(DEPT, 0)` `-- comme COALESCE, mais avec 2 arguments`
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 ***d***, 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 N*ème* occurrence de x dans y à partir de D (di D = -1, recherche droite vers gauche)	LOCATE_IN_STRING('Bonjour', 'o', 1 , 2) -> 5 --> 2ème o
INSERT(x, d, nb, ch)	insert *ch* dans x à la position *d*, en remplaçant *nb* octets (*0 admis*)	
OVERLAY(x, ch, d, [nb])	insert *ch* dans x à la position *d*, en remplaçant *nb* octets (*facultatif)*	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 *(a est l'astuce mémorisée)*	ENCRYPT_RC2(data, 'systemi', 'avant IBMi')
ENCRYPT_TDES(x, p, a)	Encrypte (algorithme TDES) x en utilisant p comme mot de passe *(a est l'astuce mémorisée)*	ENCRYPT_TDES(data, 'systemi', 'avant IBMi')
ENCRYPT_AES(x, p, a)	Encrypte (algorithme AES) x en utilisant p comme mot de passe *(a est l'astuce)*	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 *SET ENCRYPTION PASSWORD* = p)	
DECRYPT_BINARY(x, [pwd])	retourne (en binary) les données d'origine de x.(cf SET ENCRYTION PASSWORD)	
DECRYPT_CHAR(x, [pwd])	retourne (en VARCHAR) les données d'origine de x. .(cf SET ENCRYTION PASSWORD)	
VERIFY_GROUP_FOR_USER()	retourne 1 si l'utilisateur appartient à l'un des groupes indiqués, 0 sinon.	
IDENTITY_VAL_LOCAL()	retourne la dernière valeur assigné à une zone avec l'attribut AS IDENTITY (voir plutôt la clause FINAL TABLE)	

HEX(chaine)	Retourne la valeur hexa d'une chaine	`HEX('bonjour') -> 8296959196A499` `X'8296959196A499' -> Bonjour`

VARBINARY_FORMAT()	Retourne la suite binaire formattée	`varbinary_format('A7B37D20-915D-45C4-8D0B-B5AA0F864FCA',`
		`'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX')`
		` --adresse mac`
		`-> BX'A7B37D20915D45C48D0BB5AA0F864FCA`

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 d'agrégation *(fonctions de groupe)*

Fonction(x)	*Retourne ?*	*Exemple*
MAX(X)	Retourne la plus grande valeur du groupe	`MAX(DATLIV)`
MIN(X)	Retourne la plus petite valeur du groupe	`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 , **LISTAGG**(vin_nom, ',') WITHIN GROUP (order by vin_code) AS LESNOMS FROM producteur join vins Using(pr_code) **GROUP BY** pr_code, pr_nom
SYSTOOLS.SPLIT(C, 's') *	désérialise toutes les occurrences de X dans C (selon le séparateur s)	SELECT * from TABLE(systools.split(LESNOMS, ','))

ROW_NUMBER()	numérote les lignes affichées	`select ROW_NUMBER() over (), codart, libart from articles [order by un tri]`
	numérote les lignes sur un critère de tri (ici le prix)	`select ROW_NUMBER() over (order by prix), 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() over (Partition by FAM Order by prix), 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() over ([Partition by..] order by prix), 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() over ([Partition by..] order by prix), codart, libart from articles`
PERCENT_RANK()*	Retourne le % du rang (entre 0 et 1)	`select PERCENT_RANK() over ([Partition by..] order by prix), 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 décille)	`select NTILE(3) over (order by cacli ) , CACLI ,NOMCLI FROM clients -- par tiers`
CUME_DIST(X)	Retourne la distribution cumulée (le dernier vaut 1)	`select cume_dist() over (order by cacli ) , CACLI ,NOMCLI FROM clients`
FIRST_VALUE(X)	Retourne la première valeur de X suivant le tri	`select CACLI , cacli / first_value(cacli) over (order by cacli) AS NBRDEFOISPLUS FROM clients`

LAST_VALUE(X)	Retourne la dernière valeur de X suivant le tri	`select CACLI , cacli / **LAST_value(cacli)** 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 **(cacli)** over (order by cacli) AS RATIO FROM clients`

<table>
<tr>
<td><u>V6 : options pour GROUP BY</u></td>
<td>soit le SELECT basique suivant --></td>
<td>SELECT SOC, DEP, Count(*) .../...
GROUP BY soc, Dep

<table>
<tr><th>SOC</th><th>DEP</th><th>Count(*)</th></tr>
<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>
</table>
</td>
</tr>
<tr>
<td>GROUPING SETS</td>
<td>affiche les totaux pour 2 GROUPES consécutifs</td>
<td>SELECT SOC, DEP, Count(*) ...GROUP BY Grouping Sets (soc, Dep)

<table>
<tr><th>SOC</th><th>DEP</th><th>Count(*)</th></tr>
<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>
</table>
</td>
</tr>
<tr>
<td>ROLLUP</td>
<td>affiche 1 total par groupe puis des ruptures de niveau supérieur</td>
<td>SELECT SOC, DEP, Count(*) ... GROUP BY ROLLUP (soc, Dep)

<table>
<tr><th>SOC</th><th>DEP</th><th>Count(*)</th></tr>
<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>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>CUBE</td>
<td>affiche les totaux pour tous les groupes possibles</td>
<td>SELECT SOC, DEP, Count(*) GROUP BY CUBE (soc, Dep)
<table>
<tr><th>SOC</th><th>DEP</th><th>Count(*)</th></tr>
<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>
<tr><td>-</td><td>22</td><td>20</td></tr>
<tr><td>-</td><td>44</td><td>30</td></tr>
</table>
</td>
</tr>
<tr>
<td>GROUPING()</td>
<td>indique si cette ligne est le résultat de ROLLUP (rupture)</td>
<td>SELECT SOC, DEP, Count(*), GROUPING(DEP)
GROUP BY ROLLUP (soc, Dep)
<table>
<tr><th>SOC</th><th>DEP</th><th>Count(*)</th><th>Grouping(dep)</th></tr>
<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>
</table>
</td>
</tr>
</table>

7.3 : options d'agrégation	les fonctions d'agrégation peuvent être utilisées comme des fonctions OLAP (avec OVER) et sans GROUP BY	SELECT codart, prix, sum(prix) over(order by codart) from articles **CODART** / **Prix** / **SUM(prix)** **01** / **4** / **4** **02** / **15** / **19** **03** / **11** / **30** **04** / **7** / **37** **...** / **...** / **...**

Fonctions XML

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 > 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 *processing instruction*	`SELECT XMLPI(NAME "Instruction", 'APPUYEZ SUR ENTREE') FROM SYSIBM.SYSDUMMY1` `<?Instruction APPUYEZ SUR ENTREE?>`
XMLCOMMENT,	commentaire XML	`select XMLCOMMENT('A consommer avec modération') FROM SYSIBM.SYSDUMMY1 ;` `<!--A consommer avec modération→`
XMLCONCAT( )	Concatenation de deux flux XML	`select XMLCONCAT( XMLELEMENT(name "numero", nocli) , XMLELEMENT(name "nom", raisoc) ) from clients` <numero>1</numero><nom>IBM</nom> <numero>2</numero><nom>Rational</nom>
XMLFOREST( )	Suite d'éléments XML à partir des colonnes d'une table	`select XMLFOREST(nocli , raisoc) from clients` <NOCLI>1</NOCLI><RAISOC>IBM</RAISOC> <NOCLI>2</NOCLI><RAISOC>Rational</RAISOC>
XMLROW( )	arborescence XML à partir des colonnes d'une table	`select XMLROW(nocli , raisoc) from clients` <row> <NOCLI>1</NOCLI><RAISOC>IBM</RAISOC> </row> <row> <NOCLI>2</NOCLI><RAISOC>Rational</RAISOC> </row>
XMLAGG( )	Fonction d'agrégation, éléments XML par groupe (GROUP BY) ou pour la totalité du fichier (sans)	`select xmlagg(XMLELEMENT(name "nom" , raisoc)) from clients` <nom>IBM</nom><nom>Rational</nom> ....
XMLGROUP( )	Fonction d'agrégation, arborescences XML par groupe (GROUP BY) ou pour la totalité du fichier fichier (sans GROUP BY)	`select xmlgroup(XMLELEMENT(name "nom" , raisoc)) from clients` <rowset> <row><NOCLI>1</NOCLI><RAISOC>IBM</RAISOC><VILLE>New York</VILLE></row> <row><NOCLI>2</NOCLI><RAISOC>Rational</RAISOC><VILLE>Toronto</VILLE></row> </rowset>

XMLTABLE( )	Fonction Table qui traite sous forme colonnée (relationnelle), les éléments d'un flux XML. La source peut être :
	• une colonne de type XML
	• le résultat de GET_XML_FILE
	• le résultat de HTTPGETBLOB|CLOB
		`SELECT X.NOM, X.RUE, X.TEL FROM`
		` XMLTABLE ('$c/customerinfo' passing <source> as "c"`
		`  COLUMNS`
		`  NOM CHAR(30)  PATH 'name',`
		`  RUE VARCHAR(25) PATH 'addr/street',`
		`  TEL VARCHAR(20) PATH '@tel'`
		` ) AS X`

Fonctions JSON

JSON_TABLE( )	Fonction Table qui traite sous forme colonnée (relationnelle), les éléments d'un flux JSON. La source peut être :
	• une colonne de type VARCHAR
	• le résultat de GET_CLOB_FROM_FILE
	• le résultat de HTTPGETCLOB
		`SELECT X.NOM, X.RUE, X.TEL FROM`
		` JSON_TABLE (<source> , $.client[*]`
		`  COLUMNS(`
		`  NOM CHAR(30)  PATH 'lax $.name',`
		`  RUE VARCHAR(25) PATH 'lax $.addr.street',`
		`  TEL VARCHAR(20) PATH 'strict $.tel')`
		` ) AS X`

JSON_VALUE( )	retourne une valeur scalaire (unitaire)	`VALUES(JSON_VALUE( Json_VAR, '$.id' RETURNIN Integer) ==> 901;`
JSON_QUERY( )	retourne une chaîne au format JSON	`VALUES(JSON_QUERY( Json_VAR, '$.name') => {"first":"John","last:"Doe"};`

JSON_ARRAY( )	*Produit un tableau de valeur (rappel entre [et])*	`VALUES(JSON_ARRAY(`
		`(SELECT DEPTNO FROM DEPT WHERE DEPTNAME LIKE 'BRANCH OFFICE%')));`
		`==> ["F22","G22","H22","I22","J22"]`

JSON_OBJECT( )	*produit un objet JSON, chaque élément a un nom et une valeur*	`SELECT JSON_OBJECT('Nom' : LASTNAME, 'date naissance' : HIREDATE,`
		`'Salaire' SALARY) FROM EMPLOYEE WHERE EMPNO = '000020'`
		`==>{"Nom":"THOMPSON","date naissance":"1973-10-10","Salaire":41250.00}`

JSON_ARRAYAGG( )	*produit un tableau de valeurs par groupe (GROUP BY)*	`SELECT workdept, JSON_ARRAYAGG(lastname)` `FROM EMPLOYEE WHERE workdept LIKE 'D%' GROUP BY workdept;`

JSON_OBJECTAGG( )	*produit une série d'objets JSON (clé/valeur) par groupe (GROUP BY)*	`SELECT JSON_OBJECTAGG(workdept,`
		`  JSON_OBJECTAGG(char(empno) value lastname)`
		`FROM EMPLOYEE WHERE workdept LIKE 'D%' GROUP BY workdept;`

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	`EXEC SQL values cast(GET_CLOB_FROM_FILE('/monfichier.txt') as varchar(20000) ) into :variable;`

Fonctions HTTP de Systools

HTTPHEAD(url, httpheader ou chaine vide [''])	passe une requête HTTP retourne l'entête HTTP (header) retournée par le serveur
HTTPBLOB(url, GET | POST | PUT |DELETE, httpheader, [data])	passe une requête HTTP avec la méthode indiquée et retourne la réponse sous forme de BLOB
HTTPCLOB(url, GET | POST | PUT |DELETE, httpheader, [data])	passe une requête HTTP avec la méthode indiquée et retourne la réponse sous forme de CLOB
HTTPGETBLOB(url, httpheader ou chaine vide [''])	passe une requête HTTP avec la méthode GET et retourne la réponse sous forme de BLOB
HTTPGETCLOB(url, httpheader ou chaine vide [''])	passe une requête HTTP avec la méthode GET et retourne la réponse sous forme de CLOB
HTTPPOSTBLOB(url, httpheader ou chaine vide [''], data)	passe une requête HTTP avec la méthode POST et retourne la réponse sous forme de BLOB
HTTPPOSTCLOB(url, httpheader ou chaine vide [''], data)	passe une requête HTTP avec la méthode POST et retourne la réponse sous forme de CLOB

HTTPPUTBLOB(url, httpheader ou chaine vide [''], data)	passe une requête HTTP avec la méthode PUTet retourne la réponse sous forme de BLOB
HTTPPUTCLOB(url, httpheader ou chaine vide [''], data)	passe une requête HTTP avec la méthode PUTet retourne la réponse sous forme de CLOB
HTTPDELETEBLOB(url, httpheader ou chaine vide [''])	passe une requête HTTP avec la méthode DELETE et retourne la réponse sous forme de BLOB
HTTPDELETECLOB(url, httpheader ou chaine vide [''])	passe une requête HTTP avec la méthode DELETE et retourne la réponse sous forme de CLOB
URLENCODE(chaine, encodage [UTF-8 par défaut])	retourne la chaîne au format compatible avec une URL `systools.urlencode('info@volubis.fr', '') -> info%40volubis.fr`
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

```
--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('https://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;
```

```
--Toutes ces fonctions sont aussi livrées en mode "verbeux"
  (fonctions TABLE retournant le contenu et l'entête HTTP)

select * from TABLE (
 systools.httpgetclobVERBOSE('http://www.volubis.fr' , '')
) as T
+-------------------------+----------------------------------+
+    RESPONSEMSG          +     RESPONSEHTTPHEADER           +
+ <! DOCTYPE html PUBLIC..+ <httpheader responseCode="200"   +
+-------------------------+----------------------------------+
```

Expressions régulières

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')`

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.

Gestion des dates, des heures

- 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

Fonction(x)	*Retourne ?*	*Exemple*

DATE(x)
X doit être une chaîne au format SQL
(ISO fonctionne toujours)
	une date *(sur laquelle les fonctions suivantes s'appliquent)*	`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	`DAYOFWEEK(ENTRELE)`

	(1 = Dimanche, 2=Lundi, ...)	
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 *(Attention 01/01/xx donne toujours semaine 1)*	`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 1er Janv. 4712 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(datcde 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 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

	'DIFFERENCE' est la représentation caractères [char(22)] d'un écart entre deux timestamp.	
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')`

TIMESTAMP_FORMAT('c', f)	Transforme une chaine ***c*** en timestamp suivant le format ***f***
	
	f pouvant contenir :
	
	• - . / , ' : ; et (espace)
	• DD (jours) MM (mois) YY (années sur 2) YYYY (sur 4)
	• RR année ajustée (00 à 49>2000, 50 à 99>1900)
	• HH24 l'heure (24h)
	• MI les minutes
	• SS (secondes)
	• NNNNNN (micro-secondes)
		`TIMESTAMP_FORMAT('99/02/05' , 'RR/MM/DD')`
		`=1999-02-05-00.00.00.000000`
		
		le format YY/MM/DD aurait donné 2099

GENERATE_UNIQUE()	génère une valeur unique de type CHAR(13) basée sur le timestamp en cours.	`insert GENERATE_UNIQUE() ....`

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 sous-sélection globale)

```
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 (sous-sélection corrélée)

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