

SQL Trucs & Actuces

Je parcours souvent les forums « IBM i » aussi bien francophones qu'anglophones, et l'on y trouve quelque fois quelques subtilités dont je vais vous en faire profiter.

① CLAUSE ORDER BY STATIQUE

Vous faites sans aucun doute du SQL dans vos programmes. On vous a toujours conseillé de faire du SQL Statique versus Dynamique pour une question de performances. Il est difficile de se passer du Dynamique pour fabriquer notre clause WHERE à la volée et généralement de même pour la clause ORDER BY. Bob COSY nous propose de conditionner la clause ORDER BY et ainsi conserver notre requête en Statique.

```
ORDER BY  case :Tri  When 'NOM' then Concat(LastName, Firstnme)
           Else null  END ASC,
           case :Tri  When 'MAT' then Empno
           Else null  END ASC,
           case :Tri  When 'SAL' then Salary
           Else null  END DESC,
           case :Tri  When 'NUM' then Empno
           Else null  END ASC,
```

Figure 1

② Pivot SQL (V5R4)

Un internaute demande s'il est possible dans DB2 à partir de la table ci-dessous, de faire une requête afin d'obtenir plusieurs lignes dans une même colonne, c'est-à-dire un pivot.

NAME	FRUIT
Joe	Apple
Joe	Banana
Joe	Pineapple
Karl	Kiwi
Karl	Apple



NAME	FRUIT
Karl	Apple, Kiwi
Joe	Apple, Banana, Pineapple

Les Common Table Expression (V5R3), les fonctions OLAP (V5R4) ainsi que la récursivité (V5R4) toutes trois combinées nous permettent de résoudre la requête.

Commençons pas à pas, je demande d'abord un niveau de classement sur FRUIT avec une rupture par NAME en faisant appel à la fonction OLAP Row_Number()

```
with t1 as
(SELECT name, fruit,
 ROW_NUMBER()
 Over(Partition by name ORDER BY fruit) niv
 FROM mytable)
Select * from t1
```

NAME	FRUIT	NIV
Joe	Apple	1
Joe	Banana	2
Joe	Pineapple	3
Karl	Apple	1
Karl	Kiwi	2

```
with t1 as
(SELECT name, fruit,
 ROW_NUMBER()
 Over(Partition by name ORDER BY fruit) niv
 FROM mytable),
t2 (tname, tFruit, tniv) as
(select Name, Fruit, niv from t1 where niv = 1
 union all
 select Name, trim(tFruit) concat ', ' concat trim(Fruit),
 tNiv + 1
 from t1, t2
 where (name, niv) = (tname, tniv +1))
Select * from t2
```

TNAME	TFRUIT	TNIV
Joe	Apple	1
Karl	Apple	1
Joe	Apple, Banana	2
Karl	Apple, Kiwi	2
Joe	Apple, Banana, Pineapple	3

Puis je rajoute de la récursivité en prenant au départ les lignes de niveau 1 et en y ajoutant le niveau supérieur tout en concaténant les libellés.

Les deux premières lignes résultantes proviennent de la requête avec la clause « Where Niv = 1 », les autres sont issues de la récursivité.

Seulement voilà, du résultat précédent, seuls les niveaux les plus hauts m'intéressent pour chaque rupture de la colonne NAME.

Je finalise ma requête en récupérant le MAX() du niveau de chacun et en jointant le résultat pour obtenir mon libellé. (nb : un max() de libelle aurait aussi marché).

Une autre solution aurait été la création d'une UDF.

```
with t1 as
(SELECT name, fruit,
ROW_NUMBER()
Over(Partition by name ORDER BY fruit) niv
FROM mytable),
t2 (tname, tFruit, tniv) as
(select Name, Fruit, niv from t1 where niv = 1
union all
select Name, trim(tFruit) concat ', ' concat trim(Fruit),
tniv + 1
from t1, t2
where (name, niv) = (tname, tniv + 1)),
t3 as (select tName, Max(tNiv) tniv from t2 group by tName)
select t2.tName, tFruit from t2 join t3
on t2.tname = t3.tname
and t2.tniv = t3.tniv
```

TNAME	TFRUIT
Karl	Apple, Kiwi
Joe	Apple, Banana, Pineapple

③ ROLLUP, CUBE et GROUPING SETS (V6R1)

La V6R1 nous enrichie de quelques fonctions OLAP supplémentaires très intéressantes. Nous utilisons souvent les fonctions d'agrégation comme SUM, COUNT, MIN/MAX, AVG etc.. afin d'extraire des totaux de nos données. Mais quid des sous-totaux ? Ces fonctions permettent justement de calculer des sous-totaux de lignes agrégées avec la clause GROUP BY.

ROLLUP

ROLLUP permet de calculer les sous-totaux et totaux pour les colonnes que vous lui présentez selon la hiérarchie que vous lui demandez.

Les lignes marquées d'un ● représentent les sous-totaux et totaux automatiquement ajoutés par ROLLUP par rapport à un simple GROUP BY, soit un total des effectifs par service et sexe quelque soit le job, un total par service et un total général.

Pour résumer un GROUP BY ROLLUP(A, B, C) calcule les totaux suivants :

(A, B, C) → Idem GROUP BY

(A, B, null)

(A, null, null)

(null, null, null) → Total Général

```
SELECT DEPTNAME, SEX, JOB, COUNT(*) EFFECTIF
FROM TEST.EMPLOYEE E JOIN TEST.DEPARTMENT D
ON WORKDEPT = DEPTNO
WHERE DEPTNO IN ('A00', 'D21')
GROUP BY ROLLUP(DEPTNAME, SEX, JOB) |
```

DEPTNAME	SEX	JOB	EFFECTIF
ADMINISTRATION SYSTEMS	F	CLERK	2
ADMINISTRATION SYSTEMS	F	MANAGER	1
ADMINISTRATION SYSTEMS	F	-	● 3
ADMINISTRATION SYSTEMS	M	CLERK	4
ADMINISTRATION SYSTEMS	M	-	● 4
ADMINISTRATION SYSTEMS	-	-	● 7
SPIFFY COMPUTER SERVICE DIV.	F	PRES	1
SPIFFY COMPUTER SERVICE DIV.	F	SALESREP	1
SPIFFY COMPUTER SERVICE DIV.	F	-	● 2
SPIFFY COMPUTER SERVICE DIV.	M	CLERK	2
SPIFFY COMPUTER SERVICE DIV.	M	SALESREP	1
SPIFFY COMPUTER SERVICE DIV.	M	-	● 3
SPIFFY COMPUTER SERVICE DIV.	-	-	● 5
-	-	-	● 12

CUBE

CUBE va encore plus loin que ROLLUP, car il fonctionne d'une façon multidimensionnelle et renvoie toutes les combinaisons possibles de sous-totaux.

```
SELECT DEPTNAME, SEX, JOB, COUNT(*) EFFECTIF
FROM TEST.EMPLOYEE E JOIN TEST.DEPARTMENT D
ON WORKDEPT = DEPTNO
WHERE DEPTNO IN ('A00', 'D21')
GROUP BY CUBE(DEPTNAME, SEX, JOB)
```

DEPTNAME	SEX	JOB	EFFECTIF
-	F	CLERK	2
-	F	MANAGER	1
-	F	PRES	1
-	F	SALESREP	1
-	F	-	5
-	M	CLERK	6
-	M	SALESREP	1
-	M	-	7
ADMINISTRATION SYSTEMS	-	CLERK	6
SPIFFY COMPUTER SERVICE DIV.	-	CLERK	2
-	-	CLERK	8
ADMINISTRATION SYSTEMS	-	MANAGER	1
-	-	MANAGER	1
SPIFFY COMPUTER SERVICE DIV.	-	PRES	1
-	-	PRES	1
SPIFFY COMPUTER SERVICE DIV.	-	SALESREP	2
-	-	SALESREP	2

Les lignes présentées ici sont ajoutées au résultat du ROLLUP, on peut y voir le total des jobs quelque soit le sexe et/ou le département, le nombre de femmes ou d'hommes de l'entreprise etc... en résumé tous les totaux et sous-totaux croisés.

(A, B, C)
(A, B, null)
(A, null, null)
(null, null, null)
(null, B, C)
(null, B, null)
(A, null, C)
(null, null, C)

ROLLUP

CUBE

GROUPING SETS

Si l'on veut soi même choisir les sous-totaux sur les agrégats plutôt que d'obtenir tous les sous-totaux croisés comme peut l'offrir le puissant CUBE, dans ce cas il vous faudra utiliser la fonction OLAP Grouping Sets.

Attention, remarquez bien, que nous avons obtenu que les sous-totaux demandés et pas les lignes provenant d'un GROUP BY classique.

Pour ce faire il aurait fallu avoir la clause :

```
GROUPING SETS (  
(DEPTNAME, SEX, JOB),  
(DEPTNAME, SEX),  
(SEX, JOB) )
```

```
SELECT DEPTNAME, SEX, JOB, COUNT(*) EFFECTIF  
FROM TEST.EMPLOYEE E JOIN TEST.DEPARTMENT D  
ON WORKDEPT = DEPTNO  
WHERE DEPTNO IN ('A00', 'D21')  
GROUP BY  
GROUPING SETS ((DEPTNAME, SEX), (SEX, JOB)) |
```

DEPTNAME	SEX	JOB	EFFECTIF
SPIFFY COMPUTER SERVICE DIV.	F	-	2
SPIFFY COMPUTER SERVICE DIV.	M	-	3
ADMINISTRATION SYSTEMS	F	-	3
ADMINISTRATION SYSTEMS	M	-	4
-	M	SALESREP	1
-	F	PRES	1
-	F	MANAGER	1
-	M	CLERK	6
-	F	SALESREP	1
-	F	CLERK	2



On peut combiner l'ensemble de ces fonctions OLAP à la suite, du genre :
GROUP BY GROUPING SETS ((A, B), B), CUBE (C, D), ROLLUP (E, A)
Ce n'est pas interdit, mais cela peut pénaliser vos performances.

Le HAVING fonctionne de la même manière et s'applique à l'ensemble de ces fonctions.

Nouvelle fonction de colonne GROUPING()

Que vous utilisiez l'agrégat GROUP BY, ou l'une de ces trois nouvelles fonctions OLAP, vous pouvez tester si un champ possède la valeur Null (ce qui impliquerait un sous-total ou total) avec la fonction de colonne GROUPING(Nom du Champ). Exemple :

```
Select A, B, Count(*) as Nb, Grouping(A)  
FROM MaTable Group BY CUBE(A, B)
```

La fonction renverra la valeur 1 si le champ A est Null, ou 0 dans le cas contraire.

```
Select A, B, Count(*) as Nb, Grouping(A) + Grouping(B) as Niv  
FROM MaTable Group BY CUBE(A, B)
```

Dans l'exemple ci-dessus, si :

- Niv = 0 → il s'agit d'une ligne détail
- Niv = 1 → il s'agit d'un sous-total
- Niv = 2 → il s'agit du total général

④ Les nouvelles vues SYS__STAT (V5R4 & V6R1)

Le catalogue base de données « IBM i » c'est-à-dire les différentes références croisées de DB2 (QSYS2/SYS*) est enrichi à chaque nouvelle version. La V6R1 n'est pas en reste avec la livraison de vues SQL statistiques très riches en information dont certaines sont doré et déjà disponibles en V5R4 par PTF. Elles vous permettront d'obtenir des statistiques sur vos tables (fichiers) et leur contenu, index (logiques) ainsi que sur vos partitions (membres).

Nom Long	Nom Court	Version	Type de Statistiques
SYSCOLUMNSTAT	SYSCSTAT	V5R4	Sur les champs
SYSINDEXSTAT	SYSIXSTAT	V5R4	Sur les Index SQL uniquement
SYSTABLEINDEXSTAT	SYSTISTAT	V5R4	Sur les tables + Contraintes de clés + tables
SYSTABLESTAT	SYSTSTAT	V5R4	Sur les tables
SYSPACKAGESTAT	SYSPKSTAT	V6R1	Sur les *SQLPKG
SYSPROGRAMSTAT	SYSPGSTAT	V6R1	Sur les programmes contenant du SQL
SYSMQTSTAT	SYSMQTSTAT	V6R1	Sur les tables matérialisées
SYSPARTITIONSTAT	SYSPSTAT	V5R4	Même notion que les précédentes
SYSPARTITIONINDEXSTAT	SYSPISTAT	V5R4	mais cette fois-ci avec la notion de
SYSPARTITIONINDEXES	SYSPINDEX	V6R1	partitions, c'est-à-dire de membres
SYSPARTITIONMQTS	SYSPMQT	V6R1	
SYSSCHEMAS	SYSSCHEMAS	V6R1	Liste toutes les bibliothèques du système

Figure1

Quelques exemples de l'utilisation de quelques uns d'entre eux :

SYINDEXSTAT

Vous utilisez de plus en plus SQL et pour améliorer les performances, vous suivez régulièrement les recommandations de l'Advisor Index (SYSIXADV). Mais les nouveaux index créés, sont-ils réellement utilisés par le moteur SQL ? Pour le savoir, utilisons cette requête, elle nous permettra de connaître tous les index non utilisés depuis plus de 100 jours, et pourquoi nous permettre de les supprimer

```
select system_index_schema, system_index_name, LAST_USED_TIMESTAMP
from SYSINDEXSTAT where
  days(current date)
  |
  - days(ifnull(date(LAST_USED_TIMESTAMP), '01.01.2000')) > 100
and not INDEX_OWNER in ('QDIRSRV', 'QSYS')
order by 1, 2
```

SYSTEM_INDEX_SCHEMA	SYSTEM_INDEX_NAME	LAST_USED_TIMESTAMP
FRAMEWORK2	RPTCTL01	2009-06-26 00:00:00.000000
FRAMEWORK2	RPTRUN01	2009-06-26 00:00:00.000000
JEUDESSAI	XACT2	-
PAUL	XACT2	-
PTHOMAS	EL1	2009-05-19 00:00:00.000000
PTHOMAS	MYINDEX	-
PTHOMAS	QZG0000003	-
PTHOMAS	QZG0000005	-
QRYDAT04	GROSSPITIX	-
QUSRSYS	QAUGD00003	-
QUSRSYS	QZG0000001	2009-03-31 00:00:00.000000
QUSRSYS	QZG0000002	-
QUSRSYS	QZG0000003	-
QUSRSYS	QZG0000005	2009-03-31 00:00:00.000000

SYSTABLESTAT

Vos fichiers possèdent-ils beaucoup d'enregistrements supprimés ? ce qui nécessiterait un RGZPFM ! Un fichier sans « trous » est un fichier bien plus performant. Exemple ou je traque les fichiers de plus de 1.000.000 enregistrements avec au moins 10% de lignes supprimées, ou tout simplement ceux qui possèdent 80% d'espace libre.

```
with t1 as (select
SYSTEM_TABLE_SCHEMA Schema, SYSTEM_TABLE_NAME table,
dec(round( number_deleted_rows /
(cast((number_rows + number_deleted_rows)
as decimal (22, 4)))) * 100, 3), 6, 3) Pourcentage,
dec(number_rows , 11, 0) Enregs,
dec(number_deleted_rows, 11, 0) Supprimes
from SYSTABLESTAT t where number_deleted_rows > 0)
select * from t1
where (Enregs + Supprimes > 1000000 AND Pourcentage > 10)
or (Pourcentage >= 80)
Order by Enregs desc, Pourcentage desc
```

SCHEMA	TABLE	POURCENTAGE	ENREGS	SUPPRIMES
QSPL	QASPLINFO	83.077	121	594
QUSRSYS	QAPMCCCNAT	95.833	14	322
QUSRDIRDB	IBMEID0013	80.000	8	32
DB2DEMO	AFIC1	85.185	4	23
PTHOMAS	EMPLOYEE	95.238	2	40
QSYS	QADBXRIGD	100.000	0	5
DB2DEMO	CLI1AR	100.000	0	1
QUSRSYS	QAMOVAR	100.000	0	3
QUSRSYS	QASECACT	100.000	0	1

SYSPROGRAMSTAT

Vous mettez en œuvre de la sécurité par adoption de droits et utilisez USER(*OWNER). Mais les requêtes SQL dynamiques ne sont pas concernées par ce mot clé et fonctionnent plutôt avec le mot clé DYNUSRPRF(*OWNER).

Un moyen simple de recenser tous les programmes, modules ou programmes de service contenant du SQL.

```
Select (SYSTEM_PROGRAM_SCHEMA concat '/' concat SYSTEM_PROGRAM_NAME)
pgm, cast(trim(PROGRAM_TYPE ) as char(10)) type ,
cast(trim(PROGRAM_OWNER as char(10)) Owner,
DYNAMIC_USER_PROFILE DynUsrPrf, ORIGINAL_SOURCE_FILE
from SYSPROGRAMSTAT
where not left(SYSTEM_PROGRAM_SCHEMA, 1) = 'Q'
and not PROGRAM_TYPE = '*MODULE'
order by Program_Schema, Program_name
```

PGM	TYPE	OWNER	DYNUSR...	ORIGINAL_SOURCE_FILE
DB2DEMO /DEMOP37	*PGM ...	QDFTOWN	*USER	DB2DEMO/SRCDemo (DEMOP37)
DB2XTOOL /RUNIEX	*PGM ...	PTHOMAS	*USER	DB2XTOOL/QRPGLESRC (CVTDB2HTML)
DB2XTOOL /RUNXML	*PGM ...	PTHOMAS	*USER	DB2XTOOL/QRPGLESRC (CVTDB2XML)
DB2XTOOL /WRTHDRINF	*PGM ...	PTHOMAS	*USER	DB2XTOOL/QRPGLESRC (WRTHDRINF)
DB2XTOOL2 /RUNIEX	*PGM ...	QPGMR	*USER	DB2XTOOL2/QRPGLESRC (CVTDB2HTM)
DB2XTOOL2 /RUNXML	*PGM ...	QPGMR	*USER	DB2XTOOL2/QRPGLESRC (CVTDB2XML)
DB2XTOOL2 /WRTHDRINF	*PGM ...	QPGMR	*USER	DB2XTOOL2/QRPGLESRC (WRTHDRINF)
DXXSAMPLES/GENX	*PGM ...	PTHOMAS	*USER	DXXSAMPLES/QCSRC (GENX)

SYSPARTITIONINDEXES

Votre entreprise possède de nombreux fichiers logiques et/ou index et il est très coûteux au système de les maintenir au fil de l'eau. Pour un besoin particulier, vous créez un index ou LF avec plusieurs clés. En le compilant, si le système peut s'appuyer sur un index existant pour partager un chemin d'accès, il va le faire. Dans le cas contraire il faudra maintenir un nouvel index.

Exemple : un LF existe déjà avec la clé « A » que le système maintient. Vous créez un nouvel index ou LF avec les clés « A, B ». Le système va devoir maintenir ce nouveau chemin d'accès à chaque mise à jour, suppression et insertion, les performances en sont affectées. Il est recommandé dans ce cas de recompiler le premier logique, celui avec la clé « A » afin qu'il puisse partager le chemin d'accès du nouveau LF contenant les clés « A, B ».

Des logiques qui devraient être recompilés pour partager le chemin d'accès d'autres logiques ou index existent bel et bien chez vous. Comment le savoir ? SysPartitionIndexes va nous y aider :

```
-- Creation d'un fichier de travail
create table qtemp.Myfile as (
select tabschema, sys_tname Pfile, system_INDEX_NAME LFile,
TABLE_PARTITION Member,
INDEX_TYPE type,
rank() over (partition by tabschema, sys_tname, TABLE_PARTITION
            order by case INDEX_TYPE when 'INDEX ' then 1
                       when 'LOGICAL ' then 0 end,
            CREATE_TIMESTAMP) seq,
CREATE_TIMESTAMP Date, NUMBER_KEY_COLUMNS Nb_Keys, COLUMN_NAMES Keys
from SYSPindex s
Where INDEX_TYPE in ('INDEX', 'LOGICAL'))
With data;

-- Extraction depuis le fichier de travail
select distinct cast(T1.tabschema as char(15)) lib, t1.lfile
from qtemp.Myfile t1 join qtemp.Myfile t2
on t1.Tabschema = t2.tabschema
and t1.Pfile = t2.pfile
and t1.Member = t2.member
and t1.seq < t2.seq
and t1.nb_keys <= t2.nb_keys
and trim(t2.keys) like trim(t1.keys) concat '%'
and not (t2.keys) = t1.keys]
```

LIB	LFILE
PLCHECKUP	PGMREFS1F
PLCHECKUP	PGMREFS3P
QUSRSYS	QAOKL01A

Il suffit de recompiler les logiques ou recréer l'index afin que le partage ait lieu pour des performances optimales. A la recompilation, vérifiez l'historique du travail il vous dira que le chemin a été partagé.