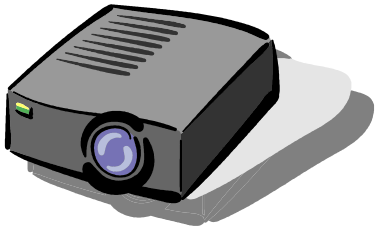
Modernisation et développement d'applications IBM i
Stratégies, technologies et outils

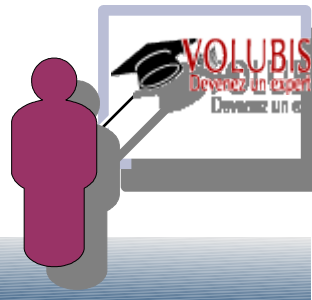
16 et 17 mai 2011 – IBM Forum de Bois-Colombes



Volubis.fr

Conseil et formation sur OS/400, I5/OS puis IBM ***i***
depuis 1994 !

Christian Massé - cmasse@volubis.fr



Nouveautés SQL.

Nouveautés liées au langage SQL en V6

- DECFLOAT nouveau format numérique 16 ou 32 octets (BigDecimal en java)
  à la norme IEEE 754R permettant le stockage de valeurs décimales très grandes :

  Jusqu'à 9.9999999999999999999999999999999999 E +/- 6144

  les valeurs INFINITY, "quiet NaN" (not a number=NAN) et "signal NaN"(SNAN)

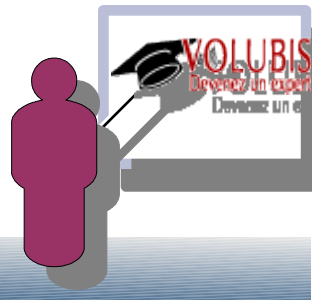
- NCHAR, NVARCHAR, NCLOB ou NATIONAL CHAR|VARCHAR|CLOB

  nouveaux types de donnée en UNICODE, UTF-8 (CCSID=1200)

- Attribut IMPLICITLY HIDDEN , lors du CREATE TABLE

  les zones cachées n'apparaissent pas en retour d'un SELECT * FROM...

  mais uniquement si vous les demandez dans la liste du SELECT



Nouveautés SQL.

Nouveautés liées au langage SQL en V6

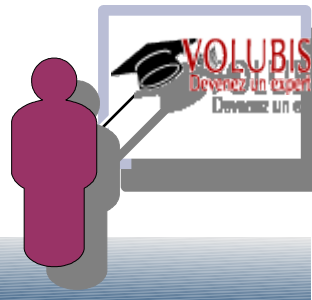
FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

attribut ajouté à une zone TIMESTAMP NOT NULL, indique que cette colonne est modifiée avec le timestamp en cours à chaque INSERT/UPDATE

Il ne peut y avoir qu'une seule colonne de ce type par table.

Vous pouvez ensuite utiliser lors de vos requêtes :

```
Select * from clients
        where ROW CHANGE TIMESTAMP FOR clients = current date - 1 day
```



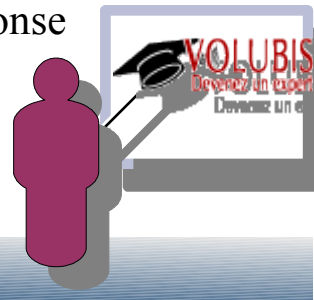
Nouveautés SQL.

Nouveautés liées au langage SQL en V6

```
ROW CHANGE TOKEN FOR nom : retourne un "token" des modifications
    indiquant si une ligne a été modifiée

Exemple : select ROW CHANGE TOKEN FOR clients into :sav_token ...
    plus tard :
        update clients set nom = 'nouveau nom' where nocli = :nocli
            and ROW CHANGE TOKEN FOR clients = :save_token
            --permet de savoir si le client a été modifié depuis lecture
```

Si la table possède une colonne FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
le TOKEN est basé sur ce Timestamp, sinon il représente un « compteur interne » de modification

→ le problème est que ce compteur interne peut être partagé par d'autres lignes (réponse officielles des labs) bref, c'est assez peu utilisable pour une table sans ROW CHANGE TIMESTAMP



Nouveautés SQL.

Nouveautés liées au langage SQL en V6

NOT LOGGED INITALLY

indiquant que la table n'est pas journalisée automatiquement

Sinon, la table est journalisée automatiquement dès la création (comme avant).

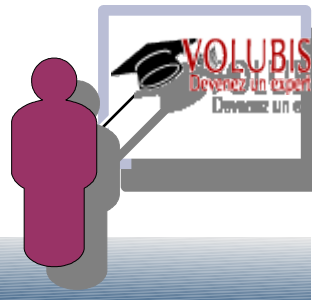
La journalisation automatique pouvant être définie par :

1/ la présence d'un journal QSQJRN

2/ la présence d'une data area QDFTJRN indiquant le nom du journal

3/ le fait d'avoir utilisé la commande STRJRNLIB sur la bibliothèque

(ce dernier point est nouveau en V6R10)



Nouveautés SQL.

Nouveautés liées au langage SQL en V6

CREATE INDEX

La création d'index subit de nombreux changements, les rendant proches des fichiers logiques (LF)

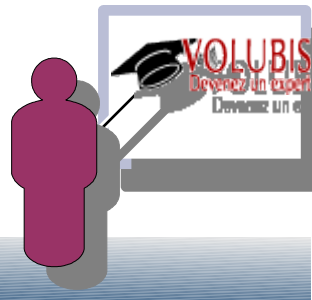
A/ On admet les expressions en tant que clé

CREATE INDEX i1 on table T1 (UPPER(NOM) as NOMMAJ)

la zone NOMMAJ est la clé de cet index.

CREATE INDEX i2 on table T1 (QTE * PRIX as MONTANT)

Vous ne pouvez pas mettre en tant que clé :
des sous requêtes
des User Defined Functions (UDF)
des fonctions agrégées (COUNT, SUM, AVG)
des fonctions NOT DETERMINISTIC
(qui ne retournent pas toujours la même valeur comme current time)



Nouveautés SQL.

Nouveautés liées au langage SQL en V6

CREATE INDEX

La création d'index subit de nombreux changements, les rendant proches des fichiers logiques (LF)

B/ On admet la clause WHERE sur les index :

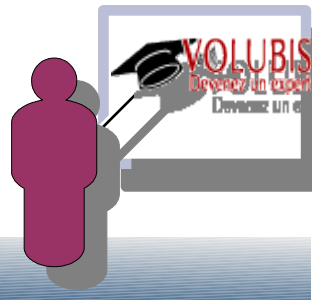
```
CREATE INDEX i3 on table T1 (NOM) WHERE DEP = 56
```

C/ Vous pouvez préciser un format par la clause RCDFMT, suivi de la phrase suivante :

ADD ALL COLUMNS : toutes les colonnes de la table appartiennent au format

ADD KEYS ONLY : seules les zones clés appartiennent au format

ADD col1, col2 : ces zones font suite aux zones clés dans le format



Nouveautés SQL.

Nouveautés liées au langage SQL en V6

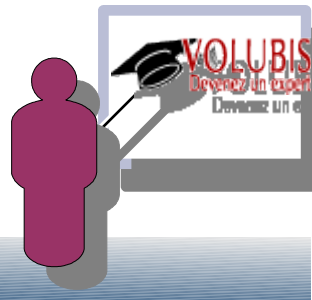
Exemple récapitulatif :

```
CREATE INDEX logi1 on HTTPLOG
    ( SUBSTR(virtualhost, 1 , 10) VHOST2 )  Where virtualhost IS NOT NULL
    RCDFMT httpfmt2 ADD host, logtime  -- en plus de VHOST2
```

L'ordre ALTER FUNCTION est nouveau et permet de modifier les attributs d'une fonction (UDF)

l'ordre LABEL ON admet en plus, à cette version :

CONSTRAINT, FUNCTION, PROCEDURE, TRIGGER et TYPE



Nouveautés SQL.

Nouveautés liées au langage SQL en V6

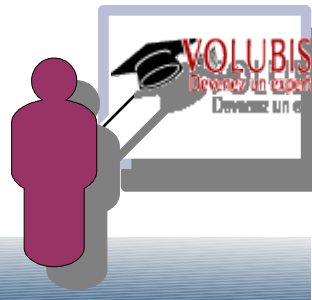
un SELECT utilisant la clause USING pour faire sa jointure :

- retourne les zones de jointure, puis les autres dans l'ordre des tables
- les zones de jointure ne doivent PAS être qualifiées

une nouvelle option de jointure est disponible: FULL OUTER JOIN

affichant l'équivalent du SELECT suivant (soit toutes les combinaisons)

```
SELECT * FROM T1 LEFT OUTER JOIN T2 on ...
 UNION
SELECT * FROM T1 RIGHT EXCEPTION JOIN T2 on ...
```



Nouveautés SQL.

Nouveautés liées au langage SQL en V6

VALUES(val1, val2, ...) peut être utilisé à la place de SELECT

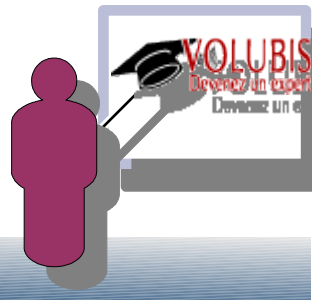
rendant possible : SELECT z1, z2, z3 FROM FICHIER WHERE ...

UNION VALUES(val1 , val2, val3)

VALUES peut être aussi utilisé simplement pour tester une fonction :

```
                          Entrée d'instructions SQL

Saisissez l'instruction SQL, puis appuyez sur ENTREE.
===> values now() + 46 days
```

```
Première ligne à afficher . . 1
....+....1....+....2....+.
VALUES
2011-05-16-16.51.02.068069
********  Fin de données  ********
```



Nouveautés SQL.

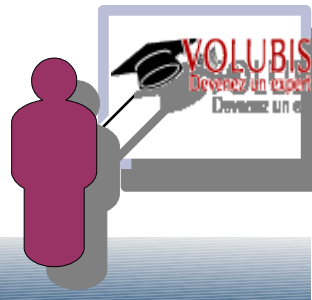
Nouveautés liées au langage SQL en V6

On peut passer un ordre SELECT sur le résultat d'un INSERT permettant ainsi, de retrouver facilement la valeur d'une zone IDENTITY ou d'un TIMESTAMP, par exemple.

Soit T1 possédant Z1 integer AS IDENTITY et Z3 de type TIMESTAMP

```
SELECT Z1, Z3 FROM FINAL TABLE
                  (INSERT INTO T1 (z2, z3) VALUES('test', now() ))

   Z1   Z3
---- ----

   3   2011-05-16-17.04.17.455674
```

Avant il fallait utiliser IDENTITY_VAL_LOCAL() pour récupérer la dernière valeur générée, toutes tables confondues.



Nouveautés SQL.

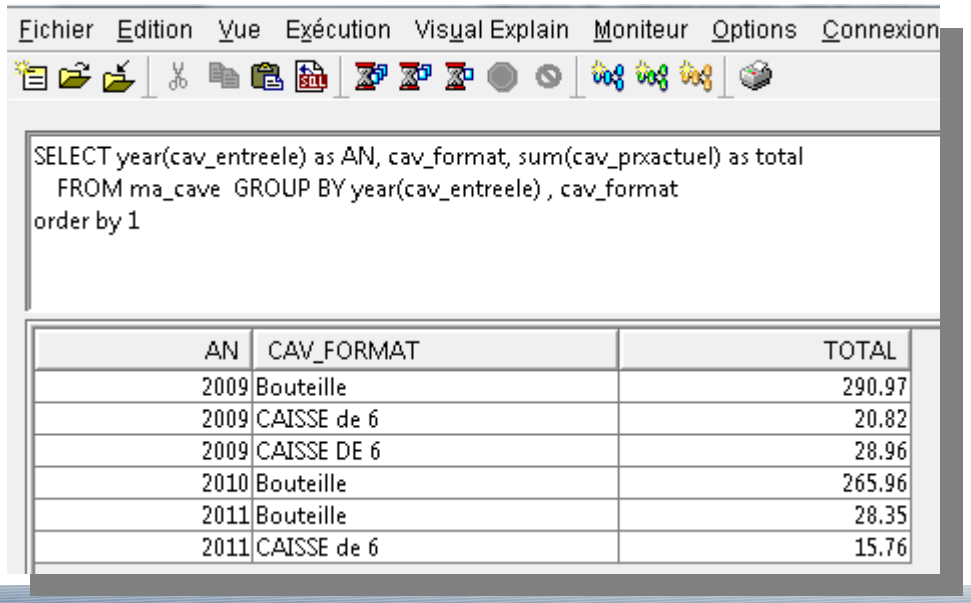
Nouveautés liées au langage SQL en V6

La clause GROUP BY évolue énormément pour implémenter des fonctions dites OLAP

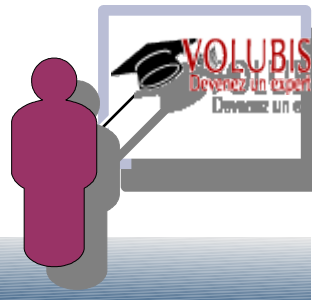
Soit la requête suivante :

```
SELECT year(cav_entreele)as AN , cav_format, sum(cav_prxactuel) as total
    FROM ma_cave GROUP BY year(cav_entreele) , cav_format order by 1
```

Affichant un total sur deux critères de « rupture »



AN	CAV_FORMAT	TOTAL
2009	Bouteille	290.97
2009	CAISSE de 6	20.82
2009	CAISSE DE 6	28.96
2010	Bouteille	265.96
2011	Bouteille	28.35
2011	CAISSE de 6	15.76



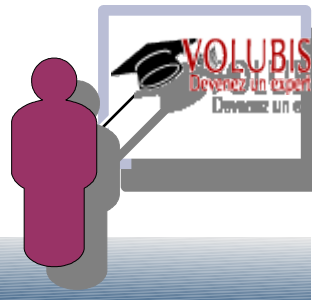
Nouveautés SQL.

Nouveautés liées au langage SQL en V6

GROUP BY GROUPING SETS ( )

```
SELECT year(cav_entreele) as AN, cav_format, sum(cav_prxactuel) as total
   FROM ma_cave  GROUP BY GROUPING SETS ( year(cav_entreele) , cav_format )
```

AN	CAV_FORMAT	TOTAL
2009	-	340.75
2010	-	265.96
2011	-	44.11
-	CAISSE DE 6 ...	28.96
-	CAISSE de 6 ...	36.58
-	Bouteille ...	585.28



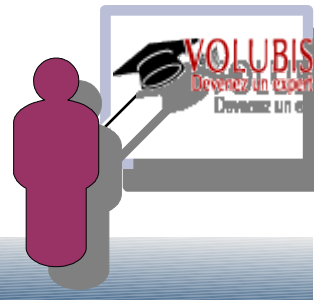
Nouveautés SQL.

Nouveautés liées au langage SQL en V6

GROUP BY GROUPING SETS (avec plusieurs groupes)

```
SELECT year(cav_entreele) as AN , MONTH(cav_entreele) as MOIS , cav_format, sum(cav_prxactuel) as TOTAL
   FROM ma_cave  GROUP BY
GROUPING SETS (  (year(cav_entreele) , cav_format) ,
                 (year(cav_entreele) , month(cav_entreele))
                 )
```

AN	MOIS	CAV_FORMAT	TOTAL
2011	-	CAISSE de 6 ...	15.76
2011	-	Bouteille ...	28.35
2009	-	CAISSE DE 6 ...	28.96
2009	-	Bouteille ...	290.97
2009	-	CAISSE de 6 ...	20.82
2010	-	Bouteille ...	265.96
2011	4	-	23.00
2011	2	-	5.35
2010	6	-	15.59
2010	4	-	74.07
2011	1	-	15.76
2009	7	-	229.38
2009	11	-	28.96
2010	1	-	20.82
2009	5	-	14.40
2010	7	-	155.48
2009	12	-	12.19
2009	2	-	20.82
2009	9	-	35.00



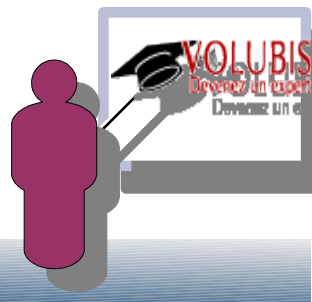
Nouveautés SQL.

Nouveautés liées au langage SQL en V6

GROUP BY ROLLUP

```
SELECT year(cav_entreele)as AN , cav_format, sum(cav_prxactuel) as total
   FROM ma_cave GROUP BY ROLLUP (year(cav_entreele) , cav_format )
```

AN	CAV_FORMAT	TOTAL
2009	Bouteille ...	290.97
2009	CAISSE de 6 ...	20.82
2009	CAISSE DE 6 ...	28.96
2009	-	340.75
2010	Bouteille ...	265.96
2010	-	265.96
2011	Bouteille ...	28.35
2011	CAISSE de 6 ...	15.76
2011	-	44.11
-	-	650.82



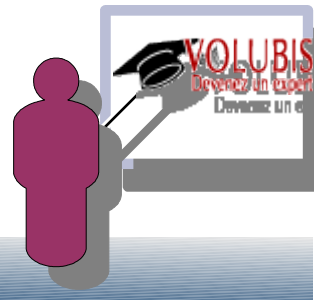
Nouveautés SQL.

Nouveautés liées au langage SQL en V6

GROUP BY CUBE

```
SELECT year(cav_entreele)as AN , cav_format, sum(cav_prxactuel) as total
   FROM ma_cave GROUP BY CUBE (year(cav_entreele) , cav_format )
```

AN	CAV_FORMAT	TOTAL
2009	Bouteille ...	290.97
2009	CAISSE de 6 ...	20.82
2009	CAISSE DE 6 ...	28.96
2009	-	340.75
2010	Bouteille ...	265.96
2010	-	265.96
2011	Bouteille ...	28.35
2011	CAISSE de 6 ...	15.76
2011	-	44.11
-	-	650.82
-	CAISSE DE 6 ...	28.96
-	CAISSE de 6 ...	36.58
-	Bouteille ...	585.28



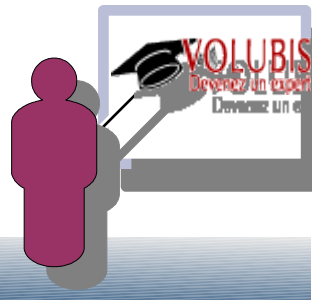
Nouveautés SQL.

Nouveautés liées au langage SQL en V6

La fonction GROUPING permettant de savoir si une ligne est une ligne de totalisation

```
SELECT year(cav_entreele)as AN , cav_format, sum(cav_prxactuel) as total,
GROUPING(cav_format)
   FROM ma_cave GROUP BY ROLLUP (year(cav_entreele) , cav_format )
```

AN	CAV_FORMAT	TOTAL	00004
2009	Bouteille ...	290.97	0
2009	CAISSE de 6 ...	20.82	0
2009	CAISSE DE 6 ...	28.96	0
2009	-	340.75	1
2010	Bouteille ...	265.96	0
2010	-	265.96	1
2011	Bouteille ...	28.35	0
2011	CAISSE de 6 ...	15.76	0
2011	-	44.11	1
-	-	650.82	1



Nouveautés SQL.

Nouveautés liées au langage SQL en V6

Autres fonctions

RID() retourne le numéro de rang en binaire (rrn est dec(15 , 0)

MONTH_BETWEEN(d1, d2) retourne le nombre de mois (avec décimales sur 31j) qui sépare les dates d1 et d2.

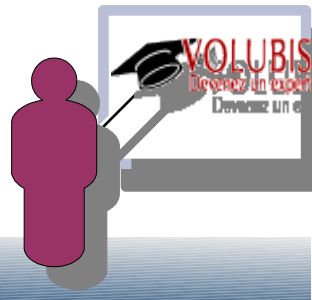
```
VALUES(months_between('25/09/08' , '31/08/08'))=> 0,806451612903225

VALUES(months_between('30/09/08' , '31/08/08'))=> 1,000000000000000
```

TIMESTAMP_FORMAT('chaîne' , 'format')

produit un Timestamp à partir de "chaîne" qui est un timestamp au format caractère, suivant le format indiqué en deuxième argument.

ROUND_TIMESTAMP(T , code) et **TRUNC_TIMESTAMP**(T , code)

fournissent un nouveau timestamp de T, arrondi ou tronqué à l'information indiqué par code *(le code pouvant représenter siècle, année, mois jusqu'à la seconde)*



Nouveautés SQL.

Nouveautés liées au langage SQL en V6

Une Sous Sélection (SELECT imbriqué dans la clause where par exemple) admet maintenant ORDER BY et FETCH FIRST x ROWS ONLY ,

permettant de retrouver plus facilement le premier ou le dernier d'un groupe

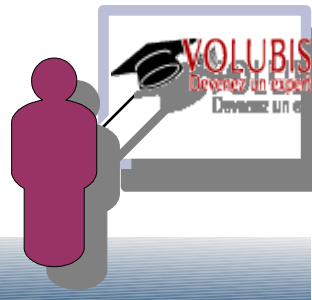
On pouvait le faire avant avec " where xxx = (SELECT MAX(xxx) from ...)

Il est possible de placer les sources SQL dans des fichiers stream,

à la compilation (CRTSQLRPGI), et au lancement d'un script (RUNSQLSTM)

avec le nouveau paramètre SRCSTMF()

```
RUNSQLSTM SRCSTMF('/home/cm/monscript.txt')
```



Nouveautés SQL.

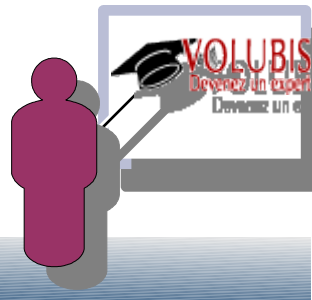
Version 7

La version 7.1 de IBM i apporte à SQL l'intégration du langage XML.

Cette intégration a été normalisée sous le nom de SQLXML, elle propose :

- Champs de type XML avec possibilité de validation d'un schéma XSD lors de l'insertion
- Fonctions pour produire du XML à partir de données élémentaires
- Sérialisation du XML en types simples (CHAR, BLOB, ...)
- Transformation du XML avec XSLT
- possibilité d'import/export avec des annotations XSD.

dans le même temps le produit OmniFind (5733OMF) est livré en V1R2
et permet l'indexation et la recherche de champs XML.



Nouveautés SQL.

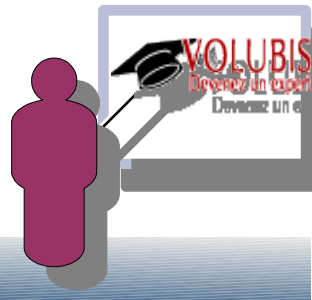
Le type XML

commençons par la création d'une table avec le type de champs XML, tel que donné par la documentation

```
CREATE SCHEMA POSAMPLE;
SET CURRENT SCHEMA POSAMPLE;
CREATE TABLE CUSTOMER (
        CID BIGINT NOT NULL PRIMARY KEY ,
        INFO XML ) ;
```

Les champs de type XML peuvent faire jusqu'à 2 Go, la totalité d'une ligne ne peut pas dépasser 3,5 Go.

Ils sont stockés dans le CCSID indiqué par SQL_XML_DATA_CCSID dans QAQQINI, UTF-8 (1208) par défaut.



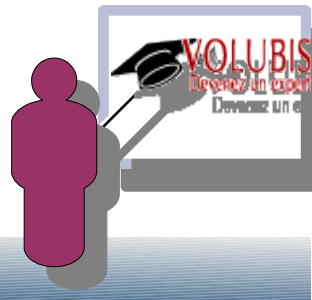
Nouveautés SQL.

Puis insertion de données, le parsing (la conversion CHAR -> XML) est automatique lors des insertions,mais vous pouvez parser de manière explicite avec XMLPARSE() pour utiliser des options (voir cette fonction)

```
INSERT INTO Customer (Cid, Info) VALUES (1000,
      '<customerinfo xmlns="http://posample.org" Cid="1000">
        <name>Kathy Smith</name>
        <addr country="Canada">
          <street>5 Rosewood</street>
          <city>Toronto</city>
          <prov-state>Ontario</prov-state>
          <pcode-zip>M6W 1E6</pcode-zip>
        </addr>
        <phone type="work">416-555-1358</phone>
       </customerinfo>')
```

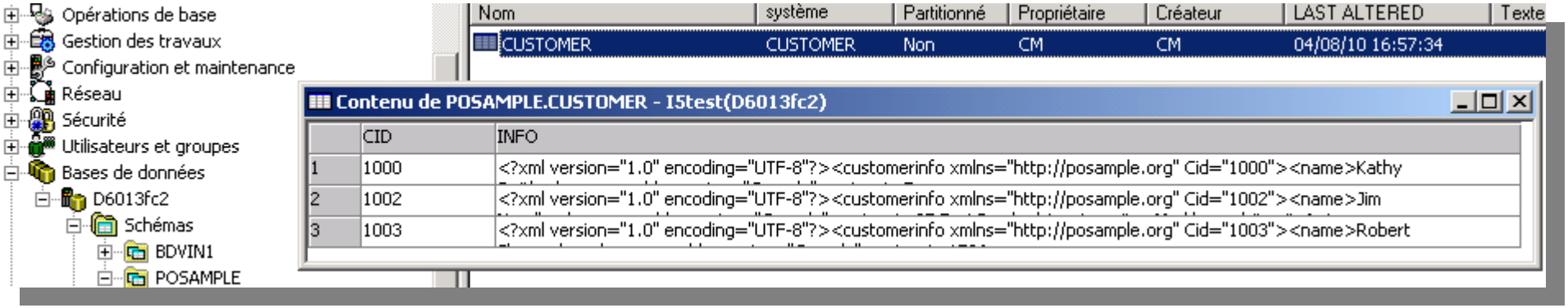
le document doit être bien formé, sinon vous recevrez SQ20398 :

→ `Echec de l'analyse syntaxique XML.`



Nouveautés SQL.

Résultat



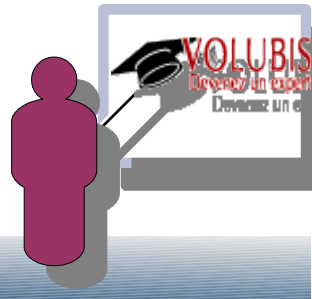
en mode 5250 la sérialisation n'est pas faite :

```
Première ligne à afficher . . _____
....+....1....+....2....+....3....+...
                      CID   INFO
                    1.000   *POINTER
                    1.002   *POINTER
                    1.003   *POINTER
********  Fin de données  ********
```

l faut utiliser XMLSERIALIZE(INFO as CHAR(2000)) pour voir le contenu,
le CCSID du job doit être renseigné (pas de CCSID à 65535).

```
Première ligne à afficher . . _____
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.
                     CID   XMLSERIALIZE
                   1.000   <customerinfo xmlns="http://posample.org" Cid="1000"><name>Kath
                   1.002   <customerinfo xmlns="http://posample.org" Cid="1002"><name>Jim
                   1.003   <customerinfo xmlns="http://posample.org" Cid="1003"><name>Robe
********  Fin de données  ********
```

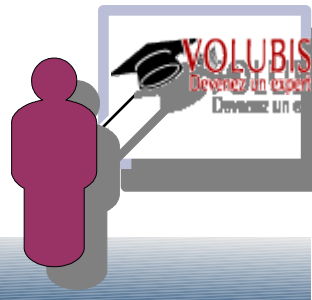
pour utiliser le type XML en programmation, voyez la session
« comment utiliser les types de données récents » demain à 11 heures



Nouveautés SQL.

Les fonctions ligne à ligne (scalaires) permettant de produire du XML :

- XMLDOCUMENT : production d'un flux XML à partir d'une chaîne de caractère (validation comprise)
- XMLPARSE : production après vérification, d'un flux XML, avec choix de conservation des espaces ou non
- XMLVALIDATE : validation d'un flux XML à l'aide d'un schéma XSD enregistré dans XSROBJECTS
- XMLTRANSFORM : transforme un flux XML à l'aide de XSLT
- XMLTEXT : production d'un texte compatible XML (*'100 est > à 99 & à 98'->'100 est > à 99 & à 98'*)
- XMLNAMESPACES , production d'un balise d'espace de nommage (namespace)
- XMLPI , production d'une balise processing instruction `(entre < ? et ?> )`
- XMLCOMMENT, production d'un commentaire XML
- XMLCONCAT, production d'un flux XML à partir de la concaténation de deux.



Nouveautés SQL.

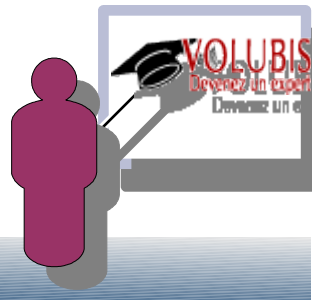
Les fonctions ligne à ligne (scalaires) permettant de produire du XML :

XMLELEMENT : production d'un élément XML

```
select  XMLELEMENT(name "region" , region) from bdvin1.regions;

 <region>Abruzzo                          </region>
 <region>Ahr                              </region>
 <region>Alsace                           </region>
 <region>Andalucía                        </region>
 <region>Aragón                           </region>
```

Cette fonction peut être imbriquée

```
select  xmlelement(name "Appellations",
          xmlelement(name "appellation" , appellation),
          xmlelement(name "region" , region_code)    )
        from bdvin1.appellations          ;

<Appellations><appellation>Alella D.O.                 </appellation><region>21</region></Appellations>
<Appellations><appellation>Ampurdàn- costa brava D.O  </appellation><region>21</region></Appellations>
<Appellations><appellation>Anoia                       </appellation><region>21</region></Appellations>
<Appellations><appellation>Bajo ebro-montsià           </appellation><region>21</region></Appellations>
<Appellations><appellation>Conca de barberà D.O        </appellation><region>21</region></Appellations>
<Appellations><appellation>Conca de tremp              </appellation><region>21</region></Appellations>
<Appellations><appellation>Costers del segre D.O       </appellation><region>21</region></Appellations>
```



Nouveautés SQL.

Les fonctions ligne à ligne (scalaires) permettant de produire du XML :

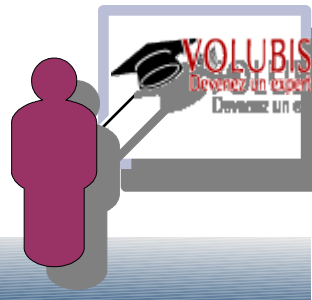
XMLATTRIBUTES : production d'un attribut, uniquement valide dans XMLELEMENT

```
select  xmlelement(name "Appellations", XMLATTRIBUTES(pays_code as "pays"),
          appellation , region_code )
 from bdvin1.appellations join bdvin1.regions using (region_code)         ;
```

```
<Appellations pays="11">Alella D.O.                21</Appellations>
<Appellations pays="11">Ampurdàn- costa brava D.O  21</Appellations>
<Appellations pays="11">Anoia                      21</Appellations>
<Appellations pays="11">Bajo ebro-montsià          21</Appellations>
```

```
select  xmlelement(name "Appellations", XMLATTRIBUTES(pays_code as "pays"),
            xmlelement(name "appellation" , appellation),
            xmlelement(name "region" , region_code)     )
from bdvin1.appellations join bdvin1.regions using (region_code)         ;
```

```
<Appellations pays="11"><appellation>Alella D.O.              </appellation><region>21</region></Appellations>
<Appellations pays="11"><appellation>Ampurdàn- costa brava D.O</appellation><region>21</region></Appellations>
<Appellations pays="11"><appellation>Anoia                    </appellation><region>21</region></Appellations>
<Appellations pays="11"><appellation>Bajo ebro-montsià        </appellation><region>21</region></Appellations>
```



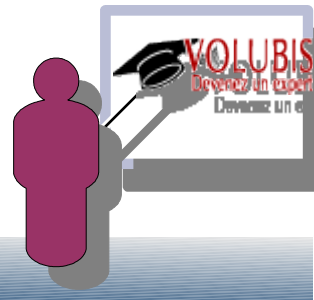
Nouveautés SQL.

Les fonctions ligne à ligne (scalaires) permettant de produire du XML :

XMLFOREST, production d'une suite d'éléments XML à partir des colonnes d'une table

```
select  XMLFOREST(region , pays_code) from bdvin1.regions;

<REGION>Abruzzo             </REGION><PAYS_CODE>18</PAYS_CODE>
<REGION>Ahr                 </REGION><PAYS_CODE>2</PAYS_CODE>
<REGION>Alsace              </REGION><PAYS_CODE>12</PAYS_CODE>
<REGION>Andalucía           </REGION><PAYS_CODE>11</PAYS_CODE>
```

```
select  XMLFOREST(region AS "region" , pays_code as "pays") from bdvin1.regions;

<region>Abruzzo             </region><pays>18</pays>
<region>Ahr                 </region><pays>2</pays>
<region>Alsace              </region><pays>12</pays>
<region>Andalucía           </region><pays>11</pays>
```



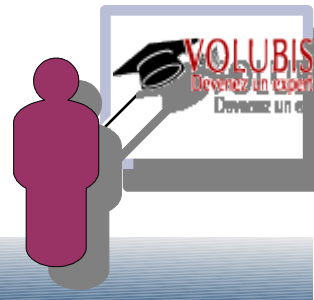
Nouveautés SQL.

Les fonctions ligne à ligne (scalaires) permettant de produire du XML :

XMLROW, production d'une suite de ligne à partir des colonnes d'une table

```
select XMLROW(appellation, region_code) from bdvin1.appellations;

<row><APPELLATION>Alella D.O.                    </APPELLATION><REGION_CODE>21</REGION_CODE></row>
<row><APPELLATION>Ampurdàn- costa brava D.O      </APPELLATION><REGION_CODE>21</REGION_CODE></row>
<row><APPELLATION>Anoia                          </APPELLATION><REGION_CODE>21</REGION_CODE></row>
<row><APPELLATION>Bajo ebro-montsià              </APPELLATION><REGION_CODE>21</REGION_CODE></row>
<row><APPELLATION>Conca de barberà D.O           </APPELLATION><REGION_CODE>21</REGION_CODE></row>
<row><APPELLATION>Conca de tremp                 </APPELLATION><REGION_CODE>21</REGION_CODE></row>
<row><APPELLATION>Costers del segre D.O          </APPELLATION><REGION_CODE>21</REGION_CODE></row>
<row><APPELLATION>Penedés D.O                    </APPELLATION><REGION_CODE>21</REGION_CODE></row>
```

```
select XMLROW(appellation, region_code OPTION ROW "une_appellation")
   from bdvin1.appellations;

<une_appellation><APPELLATION>Alella D.O.                  </APPELLATION>
    <REGION_CODE>21</REGION_CODE></une_appellation>
<une_appellation><APPELLATION>Ampurdàn- costa brava D.O    </APPELLATION>
    <REGION_CODE>21</REGION_CODE></une_appellation>
<une_appellation><APPELLATION>Anoia                        </APPELLATION>
    <REGION_CODE>21</REGION_CODE></une_appellation>
<une_appellation><APPELLATION>Bajo ebro-montsià            </APPELLATION>
    <REGION_CODE>21</REGION_CODE></une_appellation>
<une_appellation><APPELLATION>Conca de barberà D.O         </APPELLATION>
    <REGION_CODE>21</REGION_CODE></une_appellation>
<une_appellation><APPELLATION>Conca de tremp               </APPELLATION>
   <REGION_CODE>21</REGION_CODE></une_appellation>
<une_appellation><APPELLATION>Costers del segre D.O        </APPELLATION>
   <REGION_CODE>21</REGION_CODE></une_appellation>
```



Nouveautés SQL.

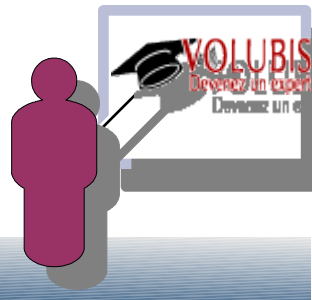
Les fonctions ligne à ligne (scalaires) permettant de produire du XML :

XMLAGG et XMLGROUP, production d'un flux XML pour un groupe de données (GROUP BY)

```
select  pays_code, xmlagg(XMLELEMENT(name "region" , region))
   from bdvin1.regions group by pays_code  ;

1  <region>Constantia            </region><region>Durbanville              </region> …
2  <region>Ahr                   </region><region>Baden                    </region> …
```

```
select pays_code, XMLGROUP(appellation, region ORDER BY region)
   from bdvin1.regions join bdvin1.appellations using(region_code)
  group by pays_code;
```

```
11 <rowset><row><APPELLATION>Aljarafe     </APPELLATION><REGION>Andalucía</REGION></row>
     <row><APPELLATION>Bailen        ...   </rowset>

12 <rowset><row><APPELLATION>Alsace       </APPELLATION><REGION>Alsace   </REGION></row>
     <row><APPELLATION>Alsace chasselas ...    </rowset>

18 <rowset><row><APPELLATION>Montepulciano</APPELLATION><REGION>Abruzo   </REGION></row>
     <row><APPELLATION>Brunelo...    </rowset>
```



Nouveautés SQL.

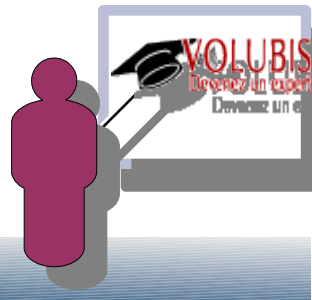
Les fonctions ligne à ligne (scalaires) permettant de produire du XML :

Schéma XSD

- Il est possible d'enregistrer un schéma dans XSROBJECTS de QSYS2 par la procédure XRS_REGISTER afin de valider les données insérées, *exemple d'insertion refusée :*

```
INSERT INTO POSAMPLE.Customer(Cid, Info) VALUES (1004,
XMLVALIDATE (XMLPARSE (DOCUMENT
         '<customerinfo xmlns="http://posample.org" Cid="1003">
           <name>Robert Shoemaker</name>
           <phone type="work">905-555-7258</phone>
           <phone type="home">416-555-2937</phone>
           <phone type="cell">905-555-8743</phone>
           <phone type="cottage">613-555-3278</phone>
         </customerinfo>' PRESERVE WHITESPACE )
  ACCORDING TO XMLSCHEMA ID posample.customer ));
```

```
--------------------------------------------------------------------------------
     Etat SQL : 2201R
     Code fournisseur : -20399
     Message : [SQ20399] Echec de l'analyse syntaxique ou de la validation XML.
      Cause . . . . . :   L'analyse syntaxique XML a échoué pendant la validation.
     Le décalage en octets dans la valeur XML en cours de traitement après conversion en UTF-8
     est de 109.La description de l'erreur de l'analyseur syntaxique XML est la suivante : cvc-
     complex-type.2.4.a:

     Expecting element with local name "addr" but saw "phone".

     Que faire . . . :   Corrigez l'incident lié au document de l'instance XML.
          Relancez XMLVALIDATE ou XDBDECOMPXML.
```



Nouveautés SQL.

Les fonctions ligne à ligne (scalaires) permettant de produire du XML :

Schéma XSD

- On peut aussi, toujours à l'aide d'un schéma XSD faire de la décomposition,

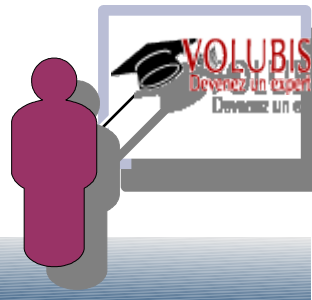
  c'est à dire "parser" le XML afin de le faire correspondre ("mapper") aux colonnes d'une table

- Dans le XSD, Ajouter l'espace de nommage db2-xdb

```
<xs:schema    xmlns:db2-xdb="http://www.ibm.com/xmlns/prod/db2/xdb1" >
```

- Définir le schéma (bibliothèque) par défaut dans une annotation

```
<xs:annotation>
    <xs:appinfo>
        <db2-xdb:defaultSQLSchema>POSAMPLE</db2-xdb:defaultSQLSchema>
    </xs:appinfo>
</xs:annotation>
```



Nouveautés SQL.

Les fonctions ligne à ligne (scalaires) permettant de produire du XML :

Schéma XSD

- Ensuite, associer à un élément ou un attribut un nom de table et de zone

```
<xs:element name="name" type="xs:string" minOccurs="1"
  db2-xdb:rowSet="CUSTOMERD" db2-xdb:column="NOM" />
```

ou

```
<xs:attribute name="country" type="xs:string"
  db2-xdb:rowSet="CUSTOMERD" db2-xdb:column="PAYS" />
```

- Puis la décomposition se fait à l'aide de la procédure XDBDECOMPXML
  qui attend un BLOB contenant un flux XML à décomposer.

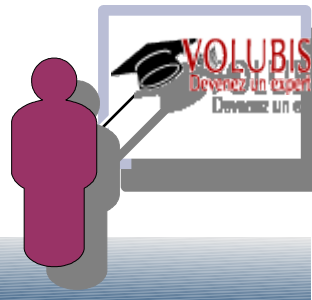
```
CALL XDBDECOMPXML ('POSAMPLE' , 'CUSTOMERD' , Get_BLOB_From_File('/tmp/decomp.xsd'), NULL);
```

résultat->

CUSTOMERD	CUSTOMERD	Non	ROOT	ROOT	12/08/10 10:44:39	

Contenu de POSAMPLE.CUSTOMERD - I5test(D6013fc2)

	NOM	RUE	VILLE	ETAT	PAYS	TEL
1	Kathy Smith	5 Rosewood	Toronto	Ontario	Canada	416-555-1358
2	Jim Noodle	1150 Maple Drive	Newtown	Ontario	Canada	905-555-7258
3	Robert Shoemaker	1596 Baseline	Aurora	Ontario	Canada	905-555-2937



Nouveautés SQL.

Le produit OmniFind (fourni gratuitement) permet une indexation

- De champs de type texte

```
select pr_avis FROM bdvin1/producteurs WHERE
        contains(pr_avis, 'chateau AND (pomerol OR lafite)') = 1  ;
```

Vous pouvez saisir

- un-mot

  en minuscules ou en majuscules, l'indexation ne tient pas compte de la casse

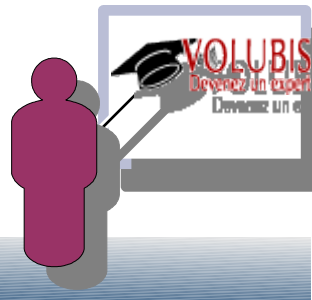
  avec ou sans le s pluriel, ainsi contains(pr_avis , 'coopérative') trouve les avis contenant coopératives

  avec sous sans les accents ainsi contains(pr_avis , 'cooperative') trouve les avis contenant coopérative(s)

  vous pouvez construire votre propre dictionnaire de synonymes

- - (tiret) un-mot

  ce mot est exclut (il ne doit pas être rencontré) contains(pr_avis , 'cave -cooperative'), recherche cave et PAS coopérative



Nouveautés SQL.

Le produit OmniFind (fourni gratuitement) permet une indexation

Vous pouvez saisir

- deux mots

  il sont implicitement reliés par AND
  `contains(pr_avis , 'cave cooperative')` trouve cave(s) ET coopérative(s)

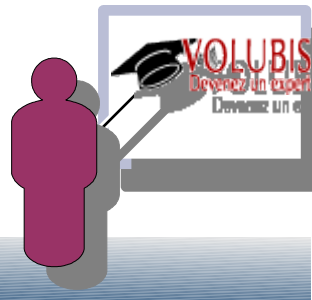
- si vous les placez entre guillemets " , c'est la chaîne qui est recherchée

  `contains(pr_avis , ' "cave cooperative" ')` trouve cave suivit de coopérative
  (les pluriels ne sont plus gérés, mais la casse et l'accentuation sont toujours ignorées)

- Opérateurs

  AND ou + , opérateur implicite
  NOT ou - , exclusion d'un mot
  OR relation OU entre deux expressions

- * caractère générique permettant d'émettre une début de mot (coop* par exemple)
- \ caractère d'échappement permettant de rechercher un + (\+) ou un - (\-) par exemple



Nouveautés SQL.

Le produit OmniFind (fourni gratuitement) permet une indexation

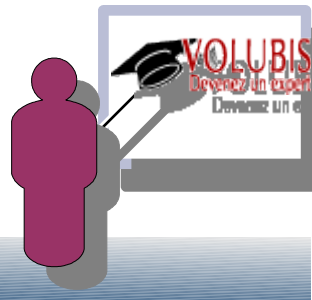
- De champs XML

  La syntaxe des recherches XML utilise un sous-ensemble du langage W3 XPath

  sous la forme CONTAINS(nom_colonne, '@xmlxp: ''expression_requête_Xpath'' ')

le deuxième paramètre de la fonction CONTAINS est une chaîne donc entre apostrophes ('), l'expression XPath étant elle même entre apostrophes il faut doubler ces dernières

```
-- liste des clients possédant un noeud "phone" = à 416-555-1358

SELECT * from posample/customer
                      where contains(info, '@xmlxp:''//phone[. = "416-555-1358"]'' ')=1


-- liste des clients possédant le noeud "phone" de customerinfo = 416-555-1358

SELECT * from posample/customer
      where contains(info,'@xmlxp:''/customerinfo[phone = "416-555-1358"]'' ')=1
```



Nouveautés SQL.

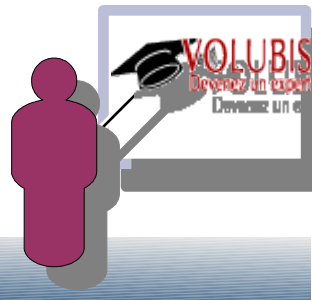
Le produit OmniFind (fourni gratuitement) permet une indexation

ON peut indexer une colonne ou le résultat produit par une fonction

Mais aussi, par le biais d'un correctif tout frais : SI40727

- Des fichiers spools
- Des fichiers de l'IFS

Peu de documentation disponible à aujourd'hui pour cette nouveauté.

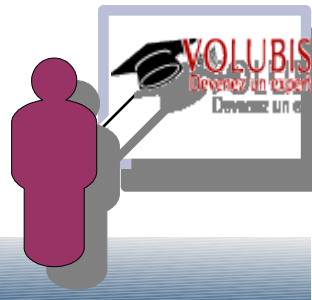


Nouveautés SQL.

Autres nouveautés V7

nouvelle instruction MERGE

```
MERGE into bdvin1/autrespays  ap
 USING( select pays_code, pays from bdvin1/pays) p
   on (ap.pays_code = p.pays_code)
 when matched then
             update set (pays_code, pays, flag) =
                        (p.pays_code, p.pays, 0)
 when not matched then
             insert (pays_code, pays, flag)
              values(p.pays_code, p.pays, 1)
```



Nouveautés SQL.

Autres nouveautés V7

Variables globales

Elles sont stockées dans des programmes de service (*SRVPGM) accessibles par toute personne ayant les droits sur l'objet. Le contenu est propre à la session .

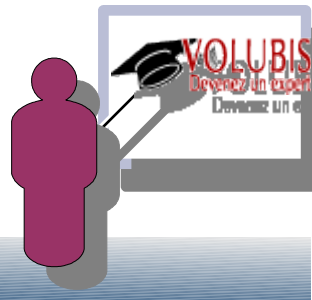
CREATE VARIABLE profil CHAR(10) DEFAULT 'QSECOFR'

la variable PROFIL sera créé pour tous les travaux du système et contiendra QSECOFR.

VALUES profil , permet de l'afficher.

VALUES 'CM' INTO PROFIL, change son contenu mais uniquement pour mon job.

la variable est initialisée en début de job, et seul le job peut la modifier.
On dit que la "portée" est limité à la session.

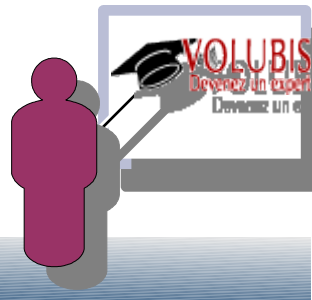


Nouveautés SQL.

Autres nouveautés V7

On peut récupérer le résultat (result sets) retourné par une procédure

```
CALL PROC03
  si SQLCODE = +446 // il y a un jeu de résultat retourné
      ASSOCIATE LOCATORS (:RS1) WITH PROCEDURE PROC03
          // :RS1 doit être déclaré SQLTYPE(RESULT_SET_LOCATOR)
     ALLOCATE C1 CURSOR FOR RESULT SET :RS1
```



Nouveautés SQL.

Autres nouveautés V7

- Nouvelles fonctions BITAND, BITANDNOT, BITOR, BITXOR et BITNOT

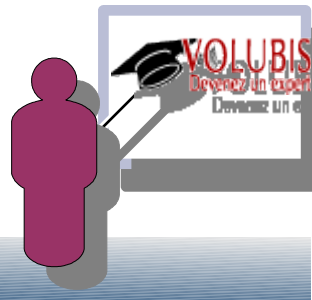
- les index EVI peuvent maintenant contenir le résultat d'une fonction agrégée (COUNT, SUM, AVG)

```
CREATE ENCODED VECTOR INDEX vinevi01
    ON  VINS (PR_CODE)

INCLUDE (COUNT(*))
```

- la procédure CANCEL_SQL de QSYS2 permet d'annuler une requête

```
CALL QSYS2.CANCEL_SQL('123456/QUSER/QZDASOINIT');
```

la PTF SI363919 implémente aussi cette procédure en V6R10



Nouveautés SQL.

Autres nouveautés V7

- SKIP LOCKED DATA | USE CURRENTLY COMMITTED | WAIT FOR OUTCOME

  La version 6 proposait déjà l'option `skip locked data permettant` d'ignorer les lignes verrouillées lors d'une lecture avec verrouillage (COMMIT(*CS) au moins).

  La version 7 amène en plus `USE CURRENTLY COMMITTED`, permettant de voir toutes les lignes, même verrouillées, mais avec les anciennes valeurs quand une transaction est en cours.

- Exemple,

  soit une mise à jour d'un prix de 33 à 35 euros sous en session 5250 avec COMMIT(*CHG)
  (l'ordre UPDATE a été lancé, pas le Commit ==> la ligne est verouillée)

  et une lecture avec COMMIT(*CS) du fichier, sous System i navigator :

  → sans paramètre (ou WAIT FOR OUTCOME), le SELECT « plante »
  → avec SKIP LOCKED DATA, on voit toutes les lignes sauf celle là (35 €)
  → avec USE CURRENTLY COMMITTED, on voit la ligne, mais à 33 euros.