Modernisation, développement d'applications et DB2 sous IBM i

Technologies, outils et nouveautés 2012-2013

8 et 9 avril 2013 – IBM Client Center Paris, Bois-Colombes

S2 – Les procédures cataloguées

Lundi 8 avril – 14h00-15h30

Pierre-Louis BERTHOIN – GAIA Mini Systèmes

Présentation

- Nous insisterons sur 3 points
  - Pourquoi utiliser des procédures stockées ?
  - Comment mettre en œuvre les procédures stockées ?
  - Comment manipuler les procédures stockées ?

Pourquoi ?

- Pour ouvrir son système d'informations en utilisant
  - Un langage standard SQL
    - Optimisation des requêtes par le moteur SQL
  - La sécurité de votre système
    - Gestion des droits utilisateurs OS
    - Souvent DBREAD, DBUPDATE
  - En récupérant des modules de votre application existante

Pourquoi ?

- Parce que c'est simple
  - A créer et à écrire en SQL
    - Langage rustique de codification
    - Commande unique de création `CREATE PROCEDURE`
  - A mettre au point
    - Plusieurs solutions

Pourquoi ?

- Pour renvoyer simplement une liste d'informations
  - sans passer par un fichier temporaire
  - `RESULT SETS`

Principe

- Procédure interne
  - Codification en SQL / PSM
    - Le langage SQL/PSM (Persistent Stored Module) est le langage procédural normalisé dans la norme SQL-99
- Le système créera un programme CLE
  - à partir du code SQL / PSM saisi

Principe

- Procédure externe
  - C’est l’utilisation d’un programme ILE
    - COBOL, RPGLE, CLLE, JAVA, etc…
- Ce programme pourra contenir du SQL embarqué

Principe

- La commande `CREATE PROCEDURE`
  - Créé un lien SQL, matérialisé par des enregistrements dans les tables
    - QSYS2/SYSROUTINES
      - Contient les procédures, fonctions et triggers
    - QSYS2/SYSPARMS
      - Contient les paramètres des procédures et fonctions
  - Pour les procédures internes ou externes

Exemples

- Nous allons voir un exemple très simple qui permet de renvoyer la commande associée à une option PDM
  - Test1
    - Procédure interne (SQL)
  - Test2
    - procédure externe
      - programme RPGLE

Exemples

- Procédure interne (création)

```
CREATE PROCEDURE TEST1 (
 IN P_OPTION CHAR(2) ,
 OUT P_COMMANDE CHAR(250) )
LANGUAGE SQL

SELECT COMMAND INTO P_COMMANDE
FROM QAUOOPT
WHERE OPTION = P_OPTION
```

- Avec une seule instruction pas besoin de

```
begin / end
```

Exemples

- Procédure externe (création)

```
CREATE PROCEDURE TEST2 (
 IN P_OPTION CHAR(2) ,
 OUT P_COMMANDE CHAR(250) )
LANGUAGE RPGLE
SPECIFIC TEST2
NOT DETERMINISTIC
READS SQL DATA
CALLED ON NULL INPUT
EXTERNAL NAME 'TEST2'
PARAMETER STYLE SQL ;
```

Exemples

- Procédure externe (programme)

```
h DFTACTGRP(*NO)

d test2           pr
d                                02
d                               250

d test2           pi
d P_option                       02
d P_command                     250

 /free
   EXEC SQL
      SELECT COMMAND INTO :P_COMMAND
      FROM QGPL/QAUOOPT
      WHERE OPTION = :P_OPTION ;
   *inlr = *on ;
 /end-free
```

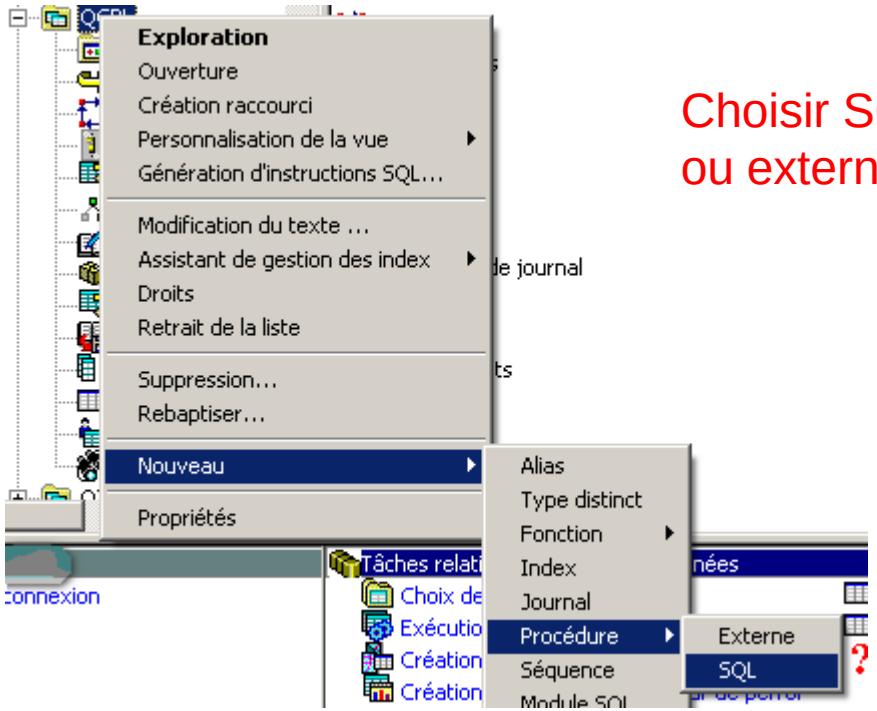
Exemples

- Procédure externe (création) pour une procédure d'un programme de service

```
CREATE PROCEDURE TEST2 (
 IN P_OPTION CHAR(2) ,
 OUT P_COMMANDE CHAR(250) )
LANGUAGE RPGLE
SPECIFIC TEST2
NOT DETERMINISTIC
READS SQL DATA
CALLED ON NULL INPUT
EXTERNAL NAME 'BERTHOIN/MYFUNCS(TEST2)'
PARAMETER STYLE SQL ;
```

Pour démarrer

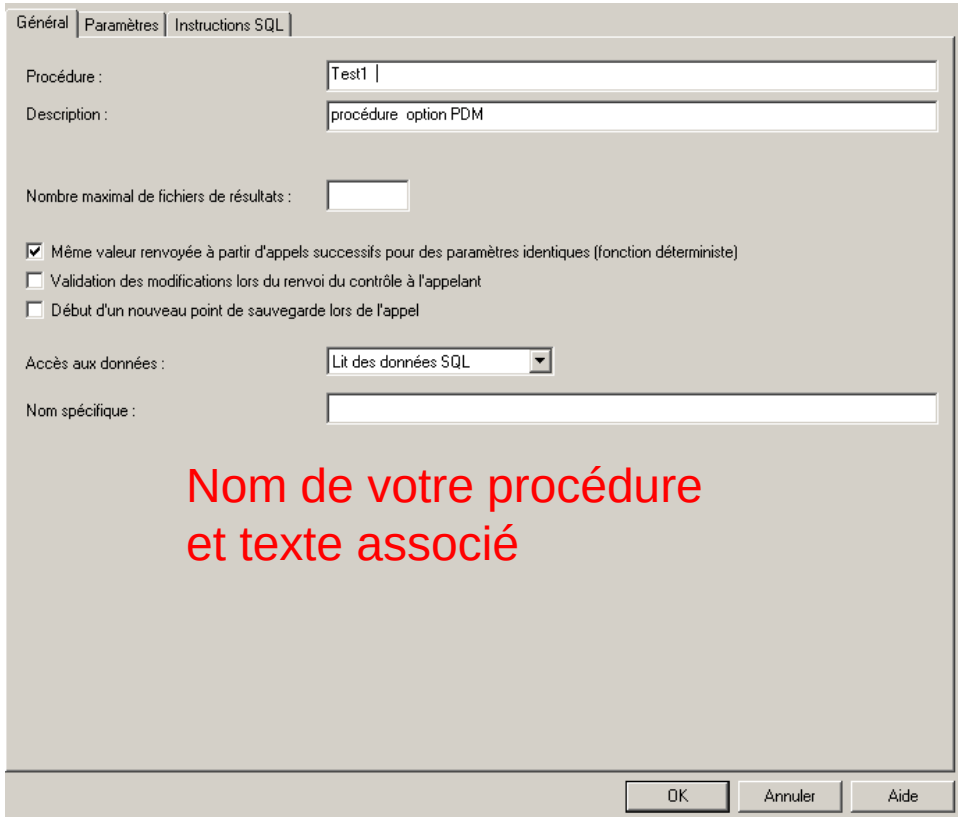
- Le plus simple : passer par Iseries navigator



Choisir SQL ou externe

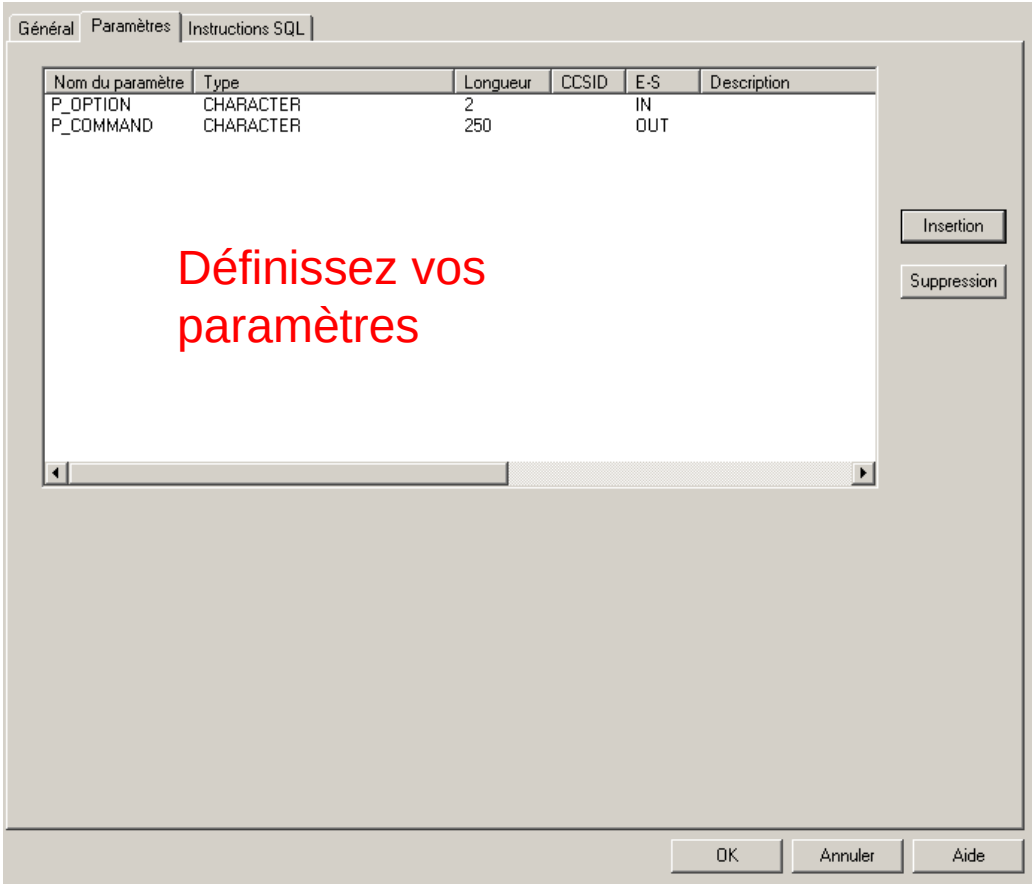
CREATE PROCEDURE (1/3)

- Informations générales



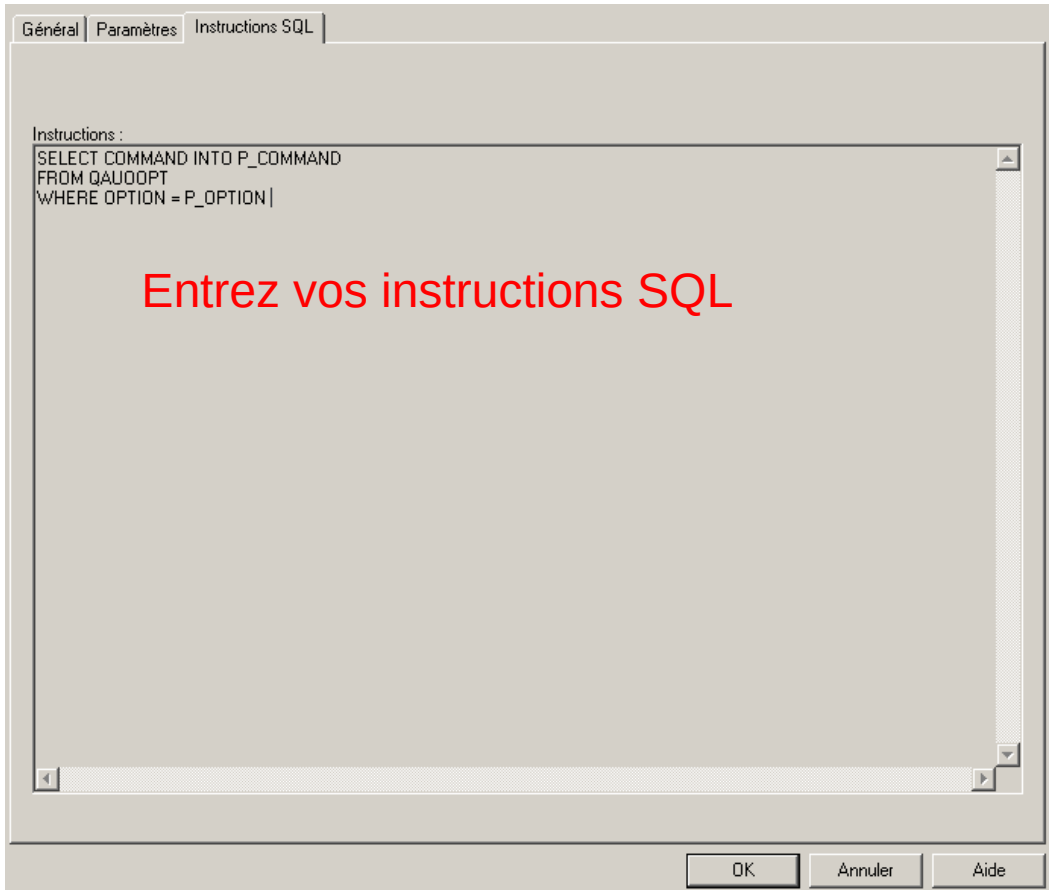
CREATE PROCEDURE (2/3)

- Paramètres



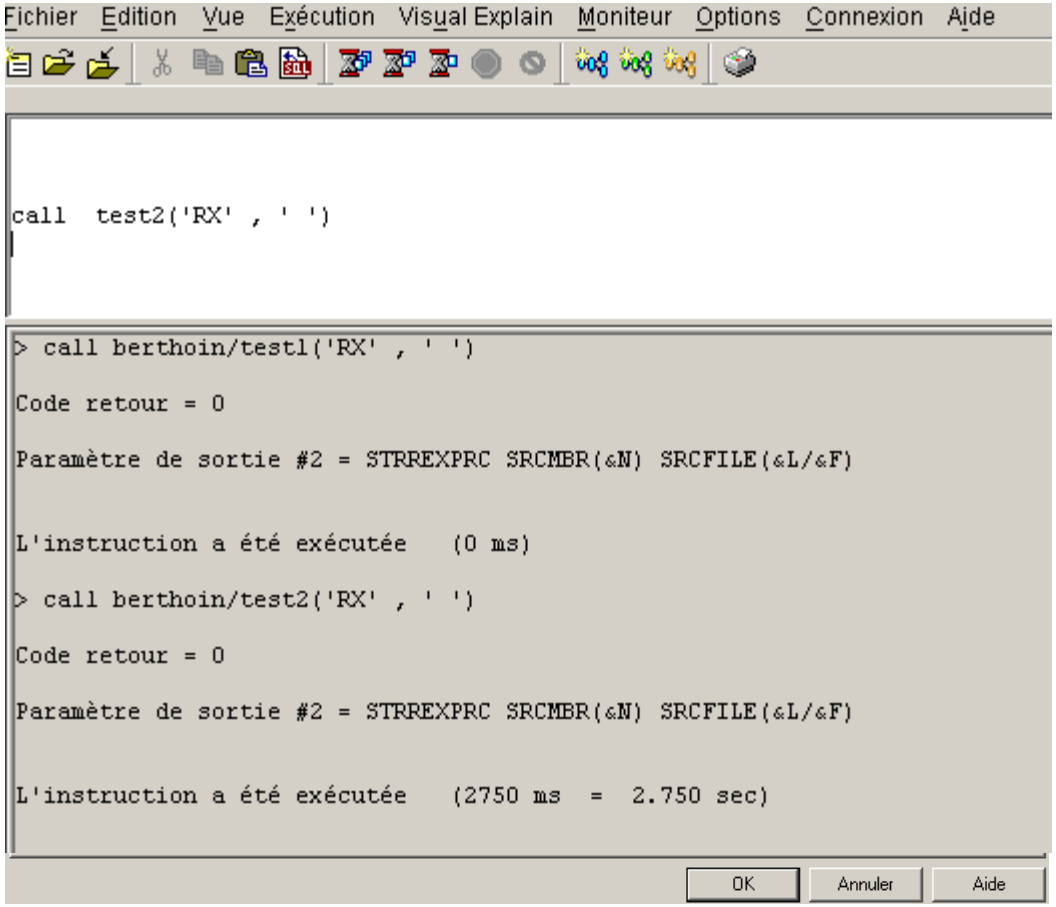
CREATE PROCEDURE (3/3)

- Corps SQL



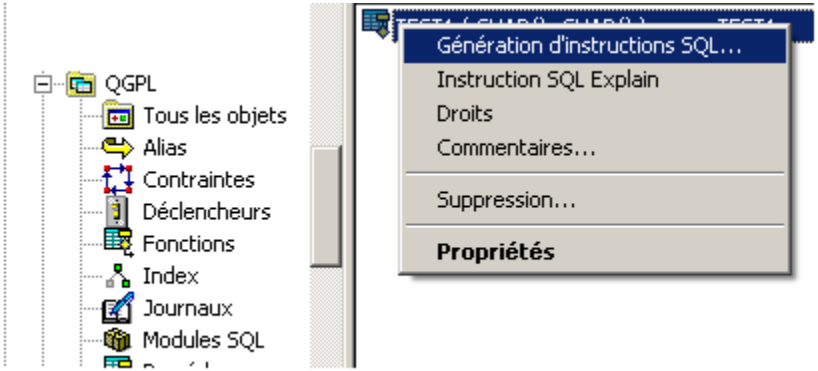
Pour tester

- Par l'exécution de script SQL

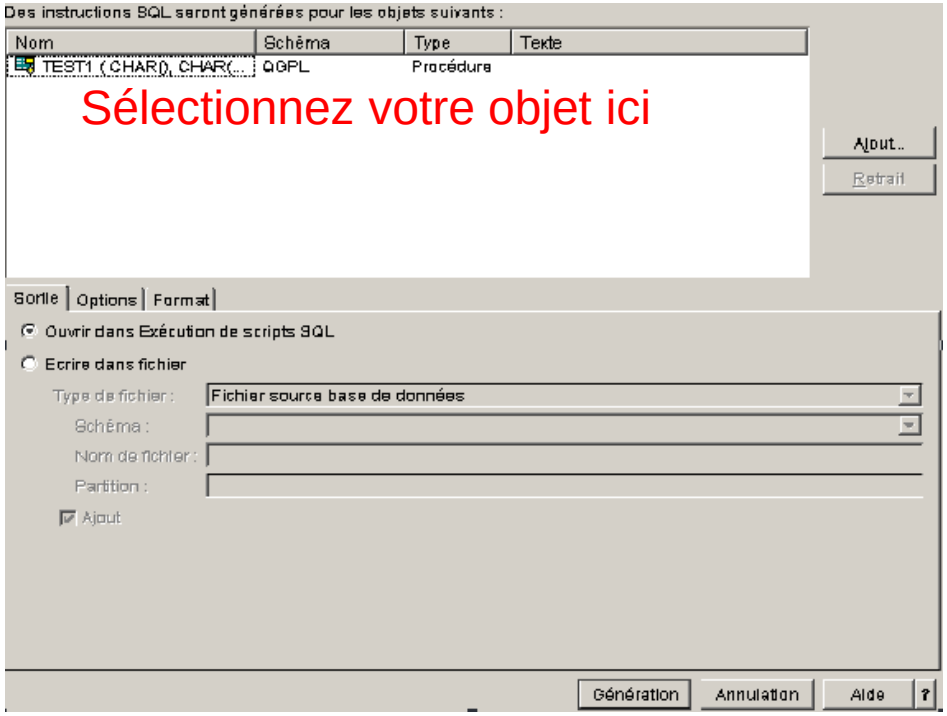


Régénération du source

- Nécessaire
  - En rétro-conception
  - En modification
  - …



Régénération du source



Pour continuer

- Vous pouvez scripter vos créations de procédure
  - dans un fichier source
  - dans l'IFS
  - vous pouvez ensuite utiliser la commande `RUNSQLSTM` pour l'exécuter

Principales options de génération

Options	Signification
NOT DETERMINISTIC DETERMINISTIC	Très important !! DETERMINISTIC => même paramètre en entrée = même résultat en sortie, il utilise son cache pour retourner l'information. Attention à la mise au point
CONTAINS SQL READS SQL DATA MODIFIES SQL DATA	Indique ce que réalise votre procédure, est-ce que son rôle est de lire des données ? faire des mises à jour ? Le moteur va vérifier que le contenu correspond à la déclaration.
CALLED ON NULL INPUT	Implique que vous devez gérer le comportement en cas input à NULL, je préfère le contraire : RETURNS NULL ON NULL INPUT
SET OPTION ALWBLK = *ALLREAD	Indique si le moteur doit verrouiller les enregistrements ou pas ; fonctionne de paire avec le niveau de contrôle transactionnel

Principales options de génération

Options	Signification
ALWCPYDTA = *OPTIMIZE	Comportement pris lorsque le système décidera s'il doit copier des données de manière temporaire. Vous pouvez lui dire que vous ne voulez pas qu'il copie des données temporaires.
COMMIT = *NONE	Isolation level : indique le comportement à adopter au niveau de l'isolation des données en cas d'accès concurrent lors de lecture et modification
DYNUSRPRF = *USER	Indique le User Profile qui sera utilisé pour les SQL dynamique
DYNAMIC RESULT SETS n	Si vous avez un result set (résultat liste)
SRTSEQ = *HEX	Mode de comparaison de chaînes

Maintenance

- Modifier une procédure
  - `ALTER PROCEDURE`
  - Peu usité
- < 7.1 : suppression et recréation
  - `DROP PROCEDURE …`
  - `CREATE PROCEDURE …`
- Depuis la 7.1
  - `CREATE **OR REPLACE** PROCEDURE …`
  - Permet la modification

Catalogage

- L'instruction CREATE PROCEDURE
  - Compile le programme CLE dans le cas d'une procédure interne
  - « Tag » le programme avec les informations SQL de catalogage
    - L'instruction CREATE PROCEDURE …
    - Permet la gestion du catalogue lors des sauvegardes, restaurations …
- PRTSQLINF

```
SELECT COMMAND INTO : H FROM QGPL / QAUOOPT WHERE OPTION = : H
  SQL4021  Dernière sauvegarde du plan d'accès le 18/03/13 à 13:34:12.
  SQL4020  La durée estimée d'exécution de la requête est de 0 secondes.
  SQL402D  Substitution attributs de requête par options de requête du fichier QAQQINI de QTEMP.
  SQL4010  Accès par ordre de table utilisé pour la table 1.
5761SS1 V6R1M0 080215     Impression d'informations SQ   Programme BERTHOIN/TEST2                  28/03/13 10:28:13  Page   002
CREATE PROCEDURE BERTHOIN . TEST2 ( IN P_OPTION CHARACTER ( 2 ) , OUT P_COMMANDE CHARACTER ( 250 ) ) READS SQL DATA PARAMETER STYL
SQL
```

Exploitation

- Avant la 7.1
  - CRTDUPOBJ, RNMOBJ
    - Sans effet sur le catalogue SQL !
  - SAV/RST
    - Sauvegarde/restaure le catalogue
  - DLTLIB
    - Supprime le catalogue pour la bibliothèque/collection
- Avec la 7.1
  - Par défaut, toutes les opérations sur le programme maintiennent le catalogue
  - Si vous aviez déjà outillé afin de palier les défauts
    - `ADDENVVAR VAR(QIBM_SQL_NO_CATALOG_UPDATE) LEVEL(*SYS)`

Langage procédural

- Structure générale

```
BEGIN
    DECLARE variable ;
    DECLARE Gestion d'erreurs
        Instructions SQL
        SET          ;
        SELECT       ;
        …
END
```

Affectation de valeurs

- SET
  - Permet d'affecter une valeur à une zone
    - Calcul
    - Castage
    - Etc …
- Exemple

```
SET CODE = 'BN';
SET Prix = QTE * PRU ;
```

Tests et alternatives

- IF

```
If test then
     instruction sql ;
     instruction sql ;
else
 instruction sql ;
End if ;
```

Tests et alternatives

- CASE

```
Case Variable
     When valeur1 then
          instruction sql ;
     When valeur2 then
          instruction sql ;
     else
          instruction sql ;
End Case ;
```

Boucles 1

- REPEAT
  - Equivalent à un DO UNTIL

```
REPEAT
    Ordre;
UNTIL
    Condition
END REPEAT
```

Boucles 2

- WHILE
  - Equivalent à un DO WHILE

```
WHILE
  condition
DO
  Ordre;
END WHILE
```

Gestion des erreurs

- Dans la procédure

```
CREATE PROCEDURE proc()
…
BEGIN

-- non trouvé
DECLARE row_not_found CONDITION FOR '02000';
DECLARE CONTINUE HANDLER FOR row_not_found
SET at_end='1';

…
If at_end …
…

END
```

Gestion des erreurs

- Pour signaler à l'appelant

```
CREATE PROCEDURE proc()
…
BEGIN

DECLARE EXIT HANDLER FOR SQLSTATE '19640'
   RESIGNAL SQLSTATE '19640'
   SET MESSAGE_TEXT ='Taille dépassée.';

IF (…)
   THEN SIGNAL SQLSTATE '19640' ;
END IF;
…

END
```

Exemple

```
CREATE PROCEDURE LSTPDM1 ( )
LANGUAGE SQL

BEGIN

DECLARE NON_TROUVE INT DEFAULT 0 ;
DECLARE MESSAGE CHAR ( 100 ) ;

DECLARE CURSOR1 CURSOR FOR
   SELECT OPTION , COMMAND FROM QAUOOPT ;

DECLARE CONTINUE HANDLER FOR NOT FOUND
SET NON_TROUVE = 1 ;

OPEN CURSOR1 ;
BOUCLE : LOOP
    FETCH FROM CURSOR1 INTO MESSAGE ;
    IF NON_TROUVE = 0 THEN
       CALL PSNDMSG ( MESSAGE ) ;
    END IF ;
END LOOP BOUCLE ;
CLOSE CURSOR1 ;

END  ;
```

Exemple

```
CREATE PROCEDURE  PSNDMSG (
  IN TEXTE_ENVOI CHAR(100) )

BEGIN

DECLARE CHANGE_CMD VARCHAR ( 256 ) ;
DECLARE CMD_LENGTH DECIMAL ( 15 , 5 ) ;
DECLARE VALEUR INTEGER ;

SET CHANGE_CMD =
'SNDMSG MSG(''' CONCAT TEXTE_ENVOI CONCAT ''') TOUSR(*REQUESTER)'
  ;
SET CMD_LENGTH = LENGTH ( CHANGE_CMD ) ;
CALL QCMDEXC ( CHANGE_CMD , CMD_LENGTH ) ;

RETURN ;

END  ;
```

RESULT SET

- Vous pouvez renvoyer une, ou plusieurs, liste(s) de valeurs
  - RESULT SET
  - Leur nombre et format ne sont pas connus à l'avance
    - Peut être dynamique (non recommandé)
- Evite de passer par une table intermédiaire
- Standard dans le monde SQL

RESULT SET SQL

```
CREATE PROCEDURE LST_PDM ( )
DYNAMIC RESULT SETS 1
LANGUAGE SQL
NOT DETERMINISTIC

BEGIN

DECLARE C1 CURSOR WITH RETURN FOR
   SELECT OPTION, COMMAND
   FROM QAUOOPT
ORDER BY OPTION ;
OPEN C1 ;

END ;
```

RESULT SET RPGLE

```
fQAUOOPT   if   e                 disk    rename( QAUOOPT : Fqauoopt )

d ResultSet       ds                        occurs( 999 )
d  FOption                          15
d  Fcommand                        250
d  Rowcount       s                  5  0

 /free

  dou %eof  ;
     read QAUOOPT ;
     if %eof ;
        leave ;
     endif ;
     RowCount = RowCount + 1;
     %Occur( ResultSet ) = RowCount ;
     foption = option ;
     fcommand = command ;
  enddo ;

  Exec SQL
     Set Result Sets Array :ResultSet for :RowCount Rows  ;
  *inlr = *On ;
/end-free
```

Nouveauté V7R1

- Lecture possible d'un result set dans un programme RPGLE, avec du SQL embarqué

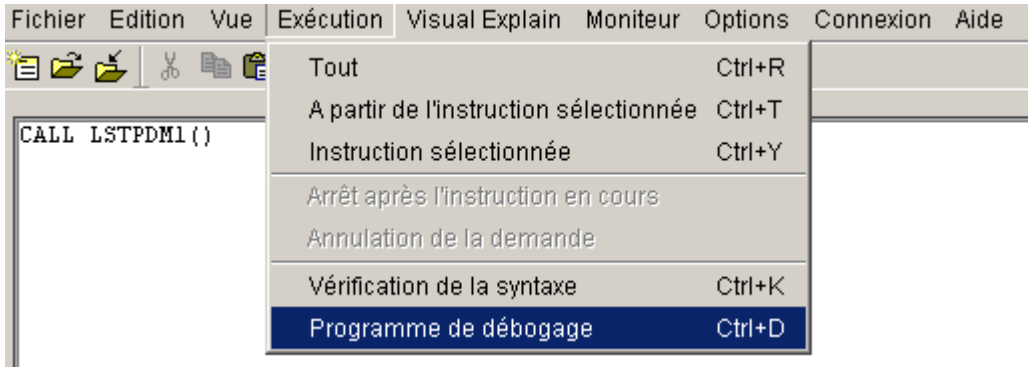
```
d MyResultSet     s                     SQLTYPE(RESULT_SET_LOCATOR)
d Row             ds                    qualified
d   Option                        2p 0
d   Commande                    250a

/free

 exec SQL call LST_PDM() ;
 exec SQL associate result set locator (:MyResultSet)
          with procedure LST_PDM ;
 exec SQL allocate C1 cursor for result set :MyResultSet ;
 exec SQL fetch next from C1 into :Row ;
 dow %subst(sqlstt:1:2) = '00' or %subst(sqlstt:1:2) = '01' ;
    // votre traitement  …
    exec SQL fetch next from C1 into :Row ;
 enddo ;
```

Mise au point

- Procédure interne (SQL)
  - Créer votre procédure avec le paramètre

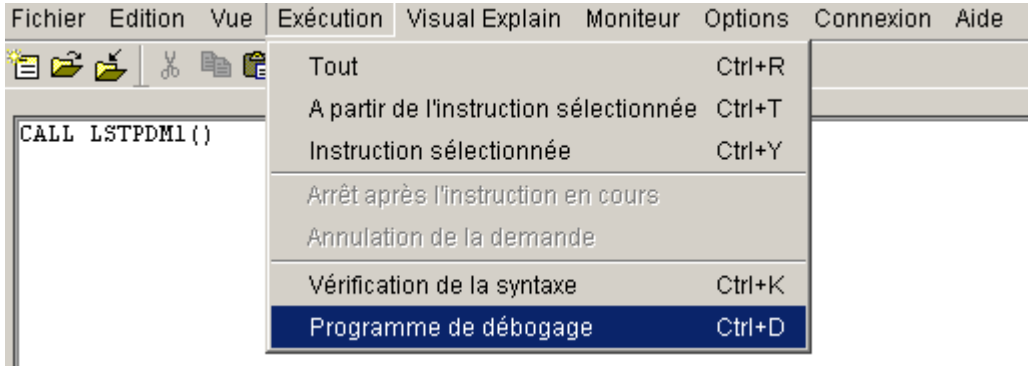
```
SET DBGVIEW = *STMT
```

- Puis par System i Navigator
  - System Debugger



Mise au point

- Procédure externe
  - Le programme doit être compilé avec la vue source
- Plusieurs débogueurs utilisables
  - STRSRVJOB + STRDBG
  - RDP
  - System Debugger



Pour aller plus loin

- Surcharge de procédures cataloguées
  - Même nom
  - Paramètres de types et/ou nombre différents
  - Chaque procédure dispose d'un `SPECIFIC NAME` unique
    - Soit indiqué par le développeur
    - Soit généré par DB2
  - Permet, par exemple
    - `CALL TO_DATE( 2013, 04, 08 ) ;`
    - `CALL TO_DATE( '2013', '04', '08' ) ;`

Informations utiles

- Ressources
  - http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0
  - http://www.redbooks.ibm.com/redbooks/pdfs/sg246503.pdf
  - Catalog management for procedures and functions
- Forums
  - http://forum.xdocs400.com
  - http://www.developpez.net
  - http://forum.commonfr.org
  - http://www.volubis.fr
  - http://www.iprodeveloper.com

Nous contacter

- Par mail
  - plberthoin@gaia.fr
  - contact@gaia.fr
- Nos sites
  - www.gaia.fr
  - www.know400.fr
  - www.as400.fr