EXPERIA
EUROPE

Lausanne 15 Juin 2010

Common Romandie

EXPERIA EUROPE
Les jardins d'Epione
38500 VOIRON
FRANCE

EXPERIA

3 activités principales :

- Editeur de logiciels systèmes (AGL **Geode**, Atelier de Traduction **ASF**, messagerie internet **INEXIA**, Interface Graphique native **SilverDev**, etc.)
- Formations (*) personnalisées sur site sur nos produits et sur l'iSeries en général (RPG IV, SQL, Sécurité, etc.)
- Prestation de service autour de nos logiciels.

EXPERIA, créée en 1991, est située à Voiron dans l'Isère.

(*) EXPERIA est organisme de formation agréé

LE RPG IV

RPG IV ou RPG ILE ?

RPG IV et RPG ILE sont une seule et même chose : ILE RPG IV

Introduction en 1994 du modèle de programmation ILE pour C/C++

Il y a donc 2 modèles :

- OPM : Original Programming Model
- ILE : Integrated Language Environnement.

RPG/400, COBOL/400, CLP, etc sont des langages OPM.

RPG IV, C/C++, COBOL Ile, CLP Ile, etc sont des langages ILE.

Pourquoi passer au RPG IV ?

… Pourquoi ne pas passer au RPG IV ?!

Les incompatibilités ? :

- **Code opération FREE non supporté**
- **Abandon de l'Autoreport (CRTRPTPGM)**

…mais qui est vraiment concerné ?

Pourquoi passer au RPG IV ?

- **Conversion** RPG III -> RPG IV **simple** (CVTRPGSRC)
- **Forte compatibilité avec le RPG III** et même le RPG
- **Simplicité d'écriture et de maintenance** (EVAL, /Free …)
- **Support des dates et arithmétique des dates**
- **Préfixes des zones de fichiers**
- **DS et Tableaux bien plus puissants** (Qualification de DS, tableaux de DS, DS dans les DS, etc …)
- **Procédure et fonctions** avec variables locales et récursivité
- **Services programmes** (« DLL »)
- **Utilisation de fonctions C et Java dans le RPG IV** !
- **Support de XML** (parser XML intégré par code opération)
- Etc, etc, etc …

3 syntaxes différentes

Le RPG IV se distingue par 3 syntaxes différentes.

- **Les spécifs C classiques**

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+Hi Lo Eq
C     *IN53         IFEQ      *ON
C     *INKC         ANDEQ     *OFF
```

3 syntaxes différentes

Le RPG IV se distingue par 3 syntaxes différentes.

- **Les spécifs C classiques**

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+Hi Lo Eq
C     *IN53          IFEQ      *ON
C     *INKC          ANDEQ     *OFF
```

- **Les spécifs C étendues**

```
CL0N01Factor1+++++++Opcode&ExtExtended_factor2++++++++++++++++++++++++++++
C                    IF        *IN53 = *ON and *INKC = *OFF
```

Facteur 1 inutilisé, absence de : zone Résultat, def de zone, indicateurs résultants. Le facteur 2 est étendu pour l'utilisation d'expressions.

Le RPG IV se distingue par 3 syntaxes différentes.

- **Les spécifs C classiques**

```
CL0N01Factor1+++++++Opcode&ExtFactor2+++++++Result++++++++Len++D+HiLoEq
C   |   *IN53          IFEQ      *ON
C   |   *INKC          ANDEQ     *OFF
```

- **Les spécifs C étendues**

```
CL0N01Factor1+++++++Opcode&ExtExtended_factor2++++++++++++++++++++++++++++
C   |                 IF        *IN53 = *ON and *INKC = *OFF
```

Facteur 1 inutilisé, absence de : zone Résultat, def de zone, indicateurs résultants. Le facteur 2 est étendu pour l'utilisation d'expressions.

- **Format libre ou /Free_**

```
/Free
   If *IN53 and not *INKC;
   ...
/End-Free
```

Les 3 syntaxes peuvent êtres mélangées dans un même source

Free et ruptures (Cycle RPG)

Pour spécifier des instructions en Free devant être exécutés en traitement total du cycle RPG (L0 à L9 ou LR), il est nécessaire de revenir en syntaxe classique de manière à indiquer le niveau.

Les instructions free suivantes seront exécutées lors des traitements totaux.

```
/Free
   // ici traitement détail
     .../...
/End-Free

 * Basculement en traitement total
CL0   L0             Tag

/Free
   if *INL1 ;
       // traitement L1
   endif;
   if *INL2;
       // traitement L1
   endif;
/End-Free
```

EVAL

Eval permet l'affectation à une variable du résultat d'une expression.

L'affectation peut être numérique ou alphanumérique (ou dates…).

Simplification de l'écriture et de la maintenance, un seul code opération en remplace plusieurs (MOVEL, Z-ADD, ADD, SUB, MULT, DIV, CAT, etc …)

```
* Exemples
C                   EVAL (H)   RES = MONT * TAUX/100

C                   EVAL       A = 'Bonjour ' + %TRIMR(Nom) +','

C                   EVAL       *IN03 = DATE1 > DATE2
```

```
 /Free
       *IN03 = Date1 > Date2;
       A = 'Bonjour ' + %TRIMR(Nom)+ ',';
       EVAL(H) Res = Mont * Taux/100;
       EVAL IN = 'ABCDE';
 /End-free
```

DATES

Les fonctions et codes opérations sur les dates permettent de simplifier les calculs sur les dates (dates d'échéances, etc…).

Exemple :

```
* Calcul n° de jour de la semaine (1=Lundi, 2=Mardi …)

D Date             S                 D
D Jour             S               10i  0

C                   Eval      jour =
C                              %rem(%Diff(d'1900-01-01':Date:*Days):7)+1
```

Ou en format libre :

```
C/Free
   Eval jour = %rem(%Diff(d'1900-01-01':Date:*Days) :7) + 1 ;

C/End-Free
```

Préfixer des champs de la BDD

Évite les rename de zones fastidieux :

```
 Fichier    Typ Design Ajout Format Type    Unité    Mots clés

FCLIENTL1     U    F       A       E      K     DISK      PREFIX('CL_')
```

Tous les champs provenant du fichier CLIENTL1 seront préfixés par 'CL_' .

Exemple : NUCLI sera connu du programme comme CL_NUCLI

Lectures/Ecriture de fichier vers/depuis une DS

Les ordres d'E/S de fichiers à description externe peuvent maintenant utiliser une DS. (Cela était disponible depuis longtemps pour les fichiers à description interne.).

Utilisation de PREFIX et/ou QUALIFIED possible sur la DS :

```
FCLIENTL1    U   F      A      E     K    DISK
...
D DS1            E  DS                  EXTNAME(CLIENTL1) Prefix(DS1_)
...
C                   Read      CLIENTL1     DS1
```

Ou en format libre :

```
C/Free
   Read ClientL1  DS1;

C/End-Free
```

Structures de données (DS)

Écriture simplifiée des sous-zones de DS : **définition par longueur.**

```
DName++++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++++
D*
D STRUCT_1        DS
D   DEBUT                         15A
D   FIN                           15A
```

Est équivalent à (définition par les positions) :

```
D STRUCT_1        DS
D   DEBUT                  1      15A
D   FIN                   16      30A
```

Les deux formes sont acceptées. Le recouvrement de zones peut se faire à l'aide du mot clé OVERLAY.

Le cadrage à gauche des noms de zones n'est plus obligatoire. L'indentation est donc possible, facilitant la lecture.

Recouvrement de tableaux

OVERLAY peut-être utilisé sur des tableaux pour créer des structures de tableaux complexes

```
DName++++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++++++++
D Struct_2         DS
D   Empl                              30A   DIM(500) ASCEND
D     Nom                             25A   OVERLAY(Empl)
D     Service                          5A   OVERLAY(Empl:26)
```

Les sous-zones nom et service étant définies par recouvrement du tableau Empl, ***elles deviennent utilisables comme des tableaux.***

Avantage : L'utilisation du tri (SORTA) sur une des sous-zones va trier tout le tableau !

```
C                   SORTA     Empl(*)
C                   SORTA     Nom(*)
C                   SORTA     Service(*)
```

DS qualifiées et tableaux de DS

```
D Addrtpl         DS
D  Rue1                          35
D  Rue2                          35
D  Cpost                          6    INZ('******')
D  Ville                         30
D  Pays                           5

D ContactTpl      DS                   Dim(10) Qualified
D  Civil                         20
D  Nom                           30
D  Prenom                        20
D  Tel                           15

D Client          DS                   Qualified
D  Name                          30
D  Addr                               Likeds(AddrTpl)
D  Contacts                           Likeds(ContactTpl)

D CliFac          DS                   LikeDs(Client)

C                   Eval      CliFac.Contacts(1).Nom = 'Dupont'
C                   Eval      CliFac.Addr.rue1 = '15,rue Victor Hugo'
C                   Eval      CliFac.Addr.rue2 = 'BP 9999'
C                   Eval      CliFac.Addr.Cpost= '12345'
```

Les procédures et fonctions

Une procédure est « une sorte » de subroutine avec les particularités suivantes :

- Paramètres (max 399)
- Hors cycle
- Récursivité (une procédure peut se rappeler elle-même)
- Variables locales (« visibles » seulement par la procédure)
- Appel externe possible (« Exportation »)
- Bibliothèque de procédures : *SRVPGM

Une fonction est une procédure qui renvoie une valeur en retour et peut s'utiliser dans une expression.

Exemple : Procédure (fonction) qui renvoie la plus petite des deux valeurs

```
...

D   Val1              S         13 2
D   Val2              S         13 2
D   Min               S         13 2

C           eval      VarX = 40
C           eval      VarY = 60
C           eval      Min = GETMIN(Val1 : Val2)

.../...

P   GETMIN        B
D                 PI          13 2     (on renvois une valeur de 13 2 )
D   VarX                      13 2     VALUE
D   VarY                      13 2     VALUE


C                 IF          VarX < VarY
C                 Return      VarX
C                 ELSE
C                 Return      VarY
C                 ENDIF

P   GETMIN        E
```

Appel de fonctions C

Le langage C (ou C++) est essentiellement constitué de procédures et fonctions dont l'implémentation sur System i est faite au travers de programmes de services (*SRVPGM)

Exemple :

CHOWN() Procédure C qui change le propriétaire d'un fichier dans l'IFS.

Prototype :

```
D   CHOWN        PR       10i0     extproc('chown')
D   Path                            *     options(*string) value
D   User                         10u0     value
D   Group                        10u0     value


C           callp       chown('/Images/Photo1.bmp' : idusr : 0)
```

Pourquoi utiliser des fonctions C ?

Les bibliothèques de fonctions C sont très riches et sans équivalents pour réaliser certaines opérations :

- Conversions de jeux de caractères (iconv..)
- Manipulations de l' « IFS » : gestion des répertoires, fichiers.
- Accès au fichiers de l'IFS : Lecture/ecriture/mise à jour du contenu de fichiers IFS (Open, Read…)
- Sockets TCP/IP : Création relativement simple de client ou serveur TCP/IP.
- Etc…

Voir le site de S. Klement : http://www.scottklement.com/rpg/

Utilisation de Java

L'appel direct de fonctions (Méthodes) écrites en JAVA est possible en RPG IV depuis la V5R1.

RPG IV permet de déclarer des objets JAVA et des méthodes (prototypage).

```
D smtpSvr             S                   O   Class(*JAVA:'java.lang.String')

D crtString           PR                  O   EXTPROC(*JAVA:
D                                             'java.lang.String':
D                                             *CONSTRUCTOR)
D                                             CLASS(*JAVA:'java.lang.String')
D                                 32000A      CONST VARYING
 /free

            smtpSvr = crtString('mail.yourwork.com');

 /end-free
```

Pourquoi utiliser JAVA ?

- De nombreux utilitaires d'utilisation libre et en Open Source sont disponibles en JAVA,
- Utilisation directe en RPG IV sans connaître JAVA

Exemples d'utilitaires :

- JavaMail : permet d'envoyer directement des eMails
- POI / HSSF : ensemble de routines Java pour manipuler des documents MS Office (essentiellement Excel). Permet de Lire, Créer et Modifier des fichiers Excel évolués (différent des simples CSV) directement depuis un programme RPG !
  (cf : http://www.scottklement.com/poi/)
- Etc

Utilisation de documents XML

Si la création documents XML est relativement simple, la lecture est très complexe.

Un analyseur de document XML est appelé un « Parser » XML.

Un parser XML (de type SAX) est intégré au RPG IV depuis la V5R4 et évolue au fil des nouvelles versions.

2 codes opérations et 2 « Bifs » permettent d'appeler le parser :

- XML-INTO
- XML-SAX
- %HANDLER
- %XML

Exemple (simple) XML-INTO

Soit le document xml `MyXMLDoc.xml` suivant :

```
<MagazineNames>
  <Name>AS/400 Magazine</Name>
  <Name>iSeries Magazine</Name>
  <Name>eServer Magazine, iSeries Edition</Name>
  <Name>IBM Systems, i5 Business Systems Edition</Name>
</MagazineNames>
```

Le code suivant permet de charger la totalité des données dans une DS

```
D progStatus       SDS
D xmlElements                    20i 0   Overlay(progStatus: 372)

D MagazineNames     DS
D Name                           50A     Dim(100) Qualified

C                   XML-Into     MagazineNames %XML('MyXMLDoc.xml')
```

La SDS permet de récupérer (en position 372) le nombre d'éléments retournés par XML-INTO (dans cet exemple, **`xmlElements`** vaudra 4)

Encore plus ...

RPG IV ILE permet également :

- Créer un programme (*PGM) à partir de sources en différents langages ILE (RPG , CL, COBOL, C)
- Appels de programmes OPM
- Allocation dynamique de mémoire
- Utilisation de pointeurs
- Fichiers « locaux » dans les procédures (V6R1/i6.1)
- etc.

Conversion RPG/400 et RPG IV : simple, mais …

Conversion simple à l'aide de CVTRPGSRC.

Mais attention aux conversions manuelles et au mélange de programmes OPM et ILE dans une même chaîne de traitement.

Contrôles supplémentaires avec EVAL :

```
C           Z-ADD   99     CTR   2 0
C           ADD     1        CTR
```

N'est pas équivalent à

```
C           Z-ADD   99    CTR   2 0
C           EVAL    CTR = CTR + 1
```

Dans le dernier cas, le EVAL provoquera une erreur à l'exécution car la variable CTR est définie en 2/0. ADD ne provoque pas d'erreur !

Conversion RPG/400 et RPG IV : Groupes d'activation

Le groupe d'activation est une sous-structure de travail.

Il permet d'allouer à un ou plusieurs programmes des ressources et d'en limiter la portée :

- Mémoire
- ODPs (Open Data Path)
- Substitutions (Overrides)
- Contrôle de validation (Commit et Rollback)

Les groupes d'activation d'un travail sont automatiquement détruits à la fin de celui-ci

Conversion RPG/400 et RPG IV : Groupes d'activation

- Tous les programmes OPM, tournent obligatoirement dans un groupe d'activation par défaut (DFTACTGRP).
- Les programmes ILE simples tournent par défaut également dans le même groupe que les OPM.
- Les programmes ILE plus complexes (utilisation de procédures, plusieurs modules …) ne peuvent pas s'exécuter dans le groupe d'activation par défaut.

PROBLEME : une chaîne de traitement mélangeant des programmes OPM et ILE dans des groupes d'activations différents peut donner des résultats imprévisibles (OVRDBF, COMMIT/ROLLBACK, … non pris en compte)

Solutions : Conversion COMPLETE de la chaîne en ILE, ou utilisation des paramètre de portée (SCOPE) des commande OVRDBF, STRCMTCTL, …

Évolutions du CLP

CLP en V5R3 et +

A partir de la V5R3 de nombreuses améliorations du CLP apparaissent :

- Définition de structures (DS)
- Nouveaux types de données (entier, pointeurs)
- Possibilités d'ouvrir jusqu'à 5 fichiers
- Nouvelles instructions de structuration
- Sub-routines - V5R4
- INCLUDE (= /COPY) – V6R1

Structures (V5R4+)

A partir de la V5R4 il est possible de définir des structures de données (équivalent de DS) à partir de la commande DCL.

Le paramètre STG (storage) doit avoir la valeur *DEFINED pour une « sous-zone ».

Le paramètre DEFVAR indique le nom de la variable de référence et la position de début.

*DCL VAR(&**AAMJ**) TYPE(*CHAR) LEN(8)*

*DCL VAR(&AN) TYPE(*CHAR) **STG(*DEFINED)** LEN(4) **DEFVAR(&AAMJ 1)***

*DCL VAR(&MOIS) TYPE(*CHAR) **STG(*DEFINED)** LEN(2) **DEFVAR(&AAMJ 5)***

*DCL VAR(&JOUR) TYPE(*CHAR) **STG(*DEFINED)** LEN(2) **DEFVAR(&AAMJ 7)***

```
 * Equivalent en RPG :
D                 DS
D AAMJ                        1       8A
D   AN                        1       4A
D   MOIS                      5       6A
D   JOUR                      7       8A
```

DCLF : 5 fichiers

5 fichiers peuvent être ouverts (en entrée et toujours un seul DSPF)

Il est nécessaire à partir de 2 fichiers de spécifier un OPNID par fichier.

Lorsqu'un OPNID est spécifié dans le DCLF , les variables programmes créées pour les champs de fichiers par le compilateur seront préfixées par cet OPNID suivi du caractère '_' (tiret bas) :

Exemple :

DCLF FILE(PCLIENT) **OPNID(CLI)**

les variables seront créées comme suit : &**CLI**_NUMCLI

L'OPNID devra être indiqué dans les commandes RCVF

A partir de la V6R1 les fichiers peuvent être explicitement fermés. Un fichier fermé par CLOSE sera automatiquement réouvert au prochain RCVF.

Instructions nouvelles

- DOWHILE
- DOUNTIL
- DOFOR
- ITERATE
- LEAVE
- SELECT/WHEN/OTHERWISE/ENDSELECT

Instructions nouvelles : exemples

```
DOUNTIL (&INDEX = 10)
   CHGVAR &INDEX (&INDEX + 1)
ENDDO
```

```
DOFOR VAR(&CTR) FROM(2) TO(4) BY(1)
 .../...
ENDDO
```

```
SELECT
WHEN (condition-1) THEN(command-1)
...
WHEN (condition-n) THEN(command-n)
OTHERWISE command-x
ENDSELECT
```

SUBR, ENDSUBR, CALLSUBR, RTNSUBR

A partir de la V5R4 il est possible de définir des subroutines.

Elles doivent être placées en fin de pgm (avant ENDPGM)

Les subroutines sont appelées par CALLSUBR. CallSubr permet de récupérer une valeur numérique en retour (RTNVAL) dans une variable CLP (ou *NONE par défaut).

La fin d'une sous routine est déterminée par le ENDSUBR ou forcée par RTNSUBR.

```
CALLSUBR SUBR(SR1) RTNVAL(&RET)
.../...
SUBR SUBR(SR1)   /* Début de sub routine */
  IF  (&A *LT 30)  THEN(RTNSUBR RTNVAL(-1))
ENDSUBR
```

CALLPRC - appel de procédure (CLP ILE)

Le CLP en ILE (ou CLLE sous PDM) permet d'appeler des procédures et fonctions exportées par d'autres modules ou des SRVPGM.

```
DCL        VAR(&SECS) TYPE(*UINT) LEN(4) VALUE(5)
DCL        VAR(&RET) TYPE(*UINT) LEN(4)

/* Appel de la fonction C : sleep() */
     CALLPRC    PRC('sleep') PARM((&SECS *BYVAL)) RTNVAL(&RET)
```

Avec CALLPRC il est ainsi possible d'appeler nos propres procédures (en RPG ou autre) mais également les fonctions C et UNIX !

INCLUDE

INCLUDE, disponible à partir de la V6R1, permet lors de la phase de compilation, d'inclure un membre source dans le source CLP en cours (équivalent de /COPY et /INCLUDE en RPG)

INCLUDE SRCMBR(membre) SRCFILE(Bib/Fichier ou *INCFILE)

*INCFILE dans SRCFILE spécifie que le membre à inclure se trouve dans le même fichier source que celui du programme (valeur par défaut).

```
INCLUDE   SRCMBR(DCLSET1)  SRCFILE(*INCFILE)
INCLUDE   SRCMBR(SUBR1)  SRCFILE(MYLIB/COMMONSUBR)
```

DCLPRCOPT – options de compilation (V6R1)

DCLPRCOPT permet de stocker dans le source d'un CLP des paramètres de compilation.

Valable en OPM et ILE (mais certains paramètres ne sont valides que dans un environnement)

Se place après PGM (dans les DCL/DCLF).

```
PGM
DCLPRCOPT    ALWRTVSRC(*NO)  USRPRF(*OWNER)
DCL  ...
```

Les valeurs indiquées dans DCLPRCOPT ont la priorité sur les options de la commande de compilation.

SQL

Insertion SQL dans un RPG :

Les instructions SQL peuvent être incorporées dans les programmes RPG (ou C, Cobol …). On parle d' « embedded SQL ».

Deux méthodes pour les SELECT :

- SQL Statique
- SQL Dynamique

Lors de la création ne pas oublier de déclarer le type de source SEU : SQLRPG ou SQLRPGLE.

Syntaxe en spécif C

L'inclusion d'instructions SQL en RPG demande une syntaxe particulière permettant le mélange des deux langages.

Le bornage des instructions SQL doit être fait pour ***chaque instruction***:

C/EXEC SQL

C/END-SQL

On va utiliser C+ pour les lignes de continuation.

Exemple :

```
C/ EXEC SQL
C+ Select * from Table1
C+ where ...
C/END-SQL
```

Exec sql en syntaxe libre

En syntaxe libre (/Free) l'inclusion de SQL est simplifié depuis la V5R4.

Les ordres SQL s'ajoutent naturellement aux autres instructions en les précédant de **exec sql** :

```
/Free
    exec sql select count(*) into :nbre_cli from Pclients ;
/end-free
```

Comme pour les instructions RPG, un ordre **exec sql** doit se terminer par un point virgule et peut se prolonger sur plusieurs ligne

Variables Hôtes

L'interaction entre le RPG et les instructions SQL se fait au moyen des « variables hôtes » (hosts).

Les variables hôtes sont des variables RPG que l'on va utiliser dans les instructions SQL en les identifiant par leur nom préfixé du caractère « : » (deux-points)

Exemple : la zone PARM1 du programme RPG peut être utilisée dans la requête mais portera le nom de : PARM1 .

Exemple :

```
C+ Select NOMCLI from Table1
C+   where NUCLI = :PARM1
```

Autre exemple :

Une instruction SELECT peut renvoyer une valeur ou une liste de valeurs qui vont alimenter des variables hôtes :

Exemple :

```
C/ EXEC SQL
C+ Select nom, fonction INTO :nom, :fonction from Table1
C+   where num = :PARM1
C/END-SQL
```

SQL Statique

La première méthode consistant à écrire le **requête en dur** dans le programme est aussi la plus simple et la plus performante.

Principales instructions SQL utilisées :

- Declare Cursor
- Open
- Fetch (lecture)
- Close

Le CURSOR

Première étape, la déclaration d'un curseur : c'est ce curseur qui va permettre l'accès aux données du ou des fichier (s) que l'on va utiliser. Un curseur est similaire à l'ODP.

DECLARE (Nom) CURSOR FOR (Requête)

Exemple :

```
C/EXEC SQL
C+    DECLARE C1 CURSOR FOR
C+       SELECT NUM, NOM, SALAIRE
C+          FROM  EMP
C+          WHERE SALAIRE > 10000
C+          ORDER BY SALAIRE ASC
C/END-EXEC
```

FETCH (“Lecture”)

L’instruction Fetch permet de lire les lignes d’un curseur ouvert.

Pour alimenter les zones RPG à partir d’un Fetch, on utilise des variables Hôtes.

> FETCH (Nom du curseur) INTO (Nom des zones avec les deux points, et séparées par une virgule.)

Il est également possible de spécifier un nom de DS comme variable Hôte.

En fin de fichier, la variable prédéfinie SQLCOD contient la valeur 100

Exemple :

```
C/EXEC SQL DECLARE C1 CURSOR FOR
C+ SELECT NUM, NOM, SALAIRE FROM  EMP
C+  WHERE SALAIRE > 10000
C+  ORDER BY SALAIRE ASC
C/END-EXEC

C/EXEC SQL
C+ OPEN C1
C/END-EXEC

C/EXEC SQL
C+ FETCH C1 INTO  :IDEMP, :NOMEMP, :SALEMP
C/END-EXEC

C/EXEC SQL
C+ CLOSE C1
C/END-EXEC
```

Exemple (RPGIV):

```
Fqprint    o    f  132        printer oflind(*in58)
D DSEMP          EDS                    EXTFILE(EMP)
C     *entry         plist
C                    parm                    salent    15 5
C/EXEC SQL DECLARE C1 CURSOR FOR
C+ SELECT NUM, NOM, SALAIRE FROM  EMP
C+  WHERE SALAIRE > :salent
C+  order by salaire asc
C/END-EXEC
C/EXEC SQL OPEN C1
C/END-EXEC
C                    except    ent
C                    DOU       sqlcod<>0
C/EXEC SQL
C+ FETCH C1 INTO  :NUM, :NOM,
C+ :SALAIRE
C/END-EXEC
C                    if        sqlcod=0
C                    except    detail
C                    endif
C   58               except    ent
C                    ENDDO
C
C
C/EXEC SQL CLOSE C1
C/END-EXEC

oqprint    e            ent         2  2
```

```
o                                          18 'Liste Employés'
o          e             ent          2
o                                           9 'N°Employé'
o                                          21 'Nom Employé'
o                                          60 'Salaire'
o          e             detail       1
o                        num                 8
o                        nom                49
o                        salaire       4    60
```

SQL Dynamique

La deuxième méthode consiste à créer la requête sous forme d'instruction « STATEMENT »

On va inclure la requête dans une chaîne de caractère ce qui va la rendre plus flexible et paramétrable (mais un peu moins performante car interprétée).

Principales instructions utilisées :

- Les mêmes qu'en SQL Statique, plus :
- Declare Statement,
- Prepare

1ère étape : déclarer l'instruction : DECLARE (Nom) STATEMENT.

C/EXEC SQL DECLARE req1 STATEMENT

2ème étape : comme en statique on déclare un curseur mais avec le nom de l'instruction : DECLARE (Nom curseur) FOR (Nom instruction)

C/EXEC SQL DECLARE C2 CURSOR FOR req1

3ème étape : Après avoir alimentée une chaîne de caractère contenant la requête, préparer l'instruction avec la chaîne de caractère correspondante : PREPARE (Nom) FROM (variable hôte)

C/EXEC SQL PREPARE req1 FROM :chaine

La suite est identique au SQL Statique.

Exemple même exercice mais avec un statement.

```
Fqprint    o    f  132        printer oflind(*in58)
D DSEMP         EDS                    EXTFILE(EMP)
D chaine          s             256
C/EXEC SQL
C+          DECLARE LECT1 STATEMENT
C/END-EXEC
C/EXEC SQL DECLARE C1 CURSOR FOR LECT1
C/END-EXEC
c                   eval       chaine='select num, nom, salaire, +
c                              comemp from emp where salaire>1000 +
c                              order by nom asc'
c

C/EXECSQL  PREPARE LECT1 FROM :chaine
C/END-EXEC
C/EXEC SQL OPEN C1
C/END-EXEC
C                   except     ent
C                   DOU        sqlcod<>0
C/EXEC SQL
C+ FETCH C1 INTO   :NUM, :NOM, :SALAIRE
C/END-EXEC
C                   if         sqlcod=0
C                   except     detail
C                   endif
C                   ENDDO
C/EXEC SQL CLOSE C1
C/END-EXEC
```

```
oqprint    e            ent          2  2
o                                          18 'Liste Employés'
o          e            ent          2
o                                           9 'N°Employé'
o                                          21 'Nom Employé'
o                                          60 'Salaire'
o                                          71 'Commission'
o          e            detail       1
o                       num                 8
o                       nom                49
o                       salaire       4    60
```

Fonctions SQL en RPG

Il est possible d'utiliser les fonctions SQL pour alimenter des variables RPG à l'aide de l'instruction SET :

Exemples

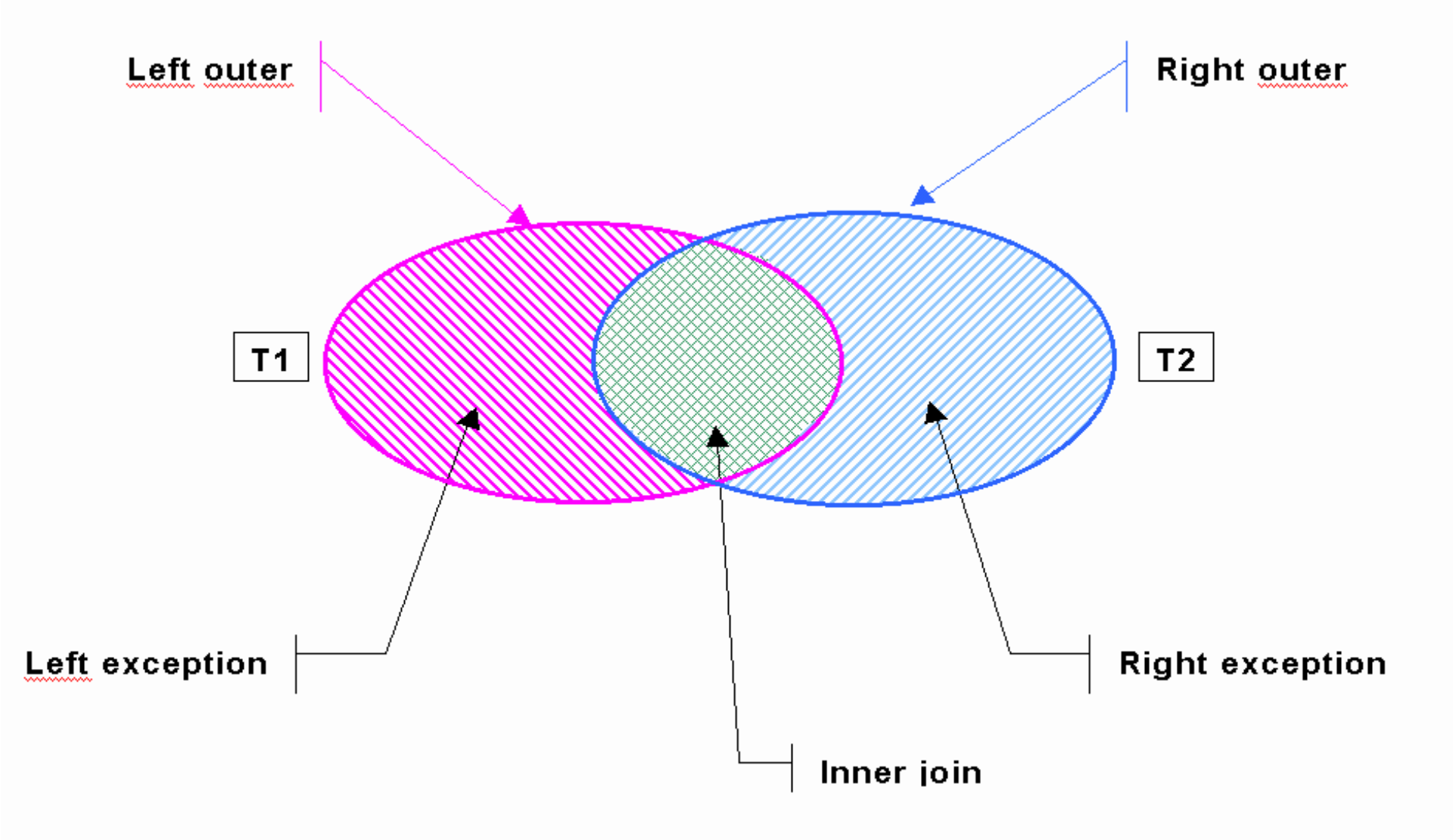
- DAYNAME() retourne le nom du jour (Lundi …) (V5R3)
- Curdate() retourne la date du jour
- WEEK_ISO() retourne le n° de semaine suivant la norme ISO

```
D Week            s               2S 0
D joursemaine     s              10A

c/EXEC SQL SET :joursemaine = DAYNAME(curdate())
c/END-EXEC

C/EXEC SQL SET :WEEK = WEEK_ISO('2010-06-15')
C/END-EXEC
```

Mémo jointures



Expression UNION

UNION permet de "joindre" des tables verticalement c'est-à-dire de **combiner dans un résultat unique des lignes provenant de deux interrogations.**

Les zone sélectionnées doivent avoir la même définition !

Donner les n° de départements (services) présent dans Emp et dans dept

Exemple :

```
SELECT n_dept FROM emp
UNION
SELECT n_dept from dept
```

Sub Select

Un SELECT peut comporter plusieurs sous-interrogations, soit imbriquées, soit au même niveau dans différents prédicats combinés par des AND ou des OR.

Exemple : Liste des employés du département 10 ayant même fonction que quelqu'un du département de « DUPONT ».

```
SELECT nom, fonction
FROM emp
WHERE n_dept = 10
AND fonction IN (SELECT fonction   FROM emp
                        WHERE n_dept =
                                (SELECT n_dept FROM emp
                                   WHERE nom = 'DUPONT'))
```

Exemple : Donner la liste des départements avec le nombre d'employés et le salaire moyen des employés du departement.

```
SELECT d.n_dept, d.nom, d.lieu, w.nbemp, w.salmoyen
FROM dept as d,
        (select n_dept, count(*) as nbemp, avg(salaire) as salmoyen
            from emp
            group by n_dept) as w
WHERE d.n_dept = w.n_dept
```

WITH

With fait partie intégrante de l'instruction SELECT. Elle permet de définir des « expressions de tables communes » (CTE : common tables expressions) à partir de sub-select , en vue de les utiliser en tant que table dans un select (ou fullselect)

WITH <nom_de_table1> (<liste de colonnes>) as <subselect> ,

<nom_de_table2> (<liste de colonnes>) as <subselect> , …

<nom_de_tableN> (<liste de colonnes>) as <subselect>

SELECT <colonnes> from nom_de_table1, nom_de_table2 ….

Le select final d'un WITH peut utiliser d'autres tables ou subselect en plus des « tables communes »

Exemple : Donner la liste des ingénieurs avec le nombre d'employés et le salaire moyen dans le département, ainsi que le salaire moyen maximum tous départements confondus

```
WITH
        INFO (DEPNUM, SALMOYEN, NBEMP) AS
              (select n_dept, avg(salaire) , count(*)
                from emp  group by n_dept),
        INFOMAX AS
              (select max(SALMOYEN) as MOYMAX
                  from INFO)
SELECT emp.*, INFO.salmoyen, INFO.nbemp, INFOMAX.moymax
   FROM emp , INFO, INFOMAX
   WHERE emp.fonction = 'Ingenieur'
        AND emp.n_dept = INFO.DEPNUM
```

Optimisations SQL : STRDBG

Pour optimiser les requêtes on peut utiliser le debug qui va « tracer » la méthode utilisée par sql pour l'accès aux fichiers.

Pour cela il faut lancer l'instruction : STRDBG *avant* d'exécuter une requête (console, programme …)

Le joblog du travail donne des informations et des conseils (création d'index, etc.) de SQL

Attention :

Dans certains cas SQL préfère créer son propre index plutôt qu'utiliser un index existant.

Cela peut-être le cas notamment quand le fichier concerné contient peu d'enregistrements.

Optimisations SQL : CQE et SQE

Depuis la version V5R2, SQL intègre 2 moteurs de requêtes :

- CQE (Classic Query Engine)
- SQE (SQL Query Engine)

Un « dispatcher » est chargé de transférer les requêtes à l'un ou l'autre des moteur.

CQE est l'ancien moteur de requête pour : OPNQRYF, Query et SQL.

SQE est un moteur de requêtes purement SQL, plus performant que CQE.

Cependant, des requêtes SQL peuvent être transférées à CQE dans certains cas. Notamment :

- requêtes sur fichiers logiques avec RENAME ou des sélections/omissions *(Obsolète à partir de la V6)*

Il est conseillé de faire les requêtes directement sur les physiques.

Optimisations SQL : INDEX

L'optimiseur de requête s'appuie sur les index pour fournir le résultat le plus rapidement possible. Pour SQL un index est :

- Soit un logique (LF) avec clés créé par des DDS
- Soit un index purement SQL (Devient un LF)

Mais : ne jamais indiquer un LF dans la clause FROM d'un select (sauf si c'est une Vue SQL) car cela force l'utilisation de CQE.

Privilégier les index créés par SQL (Create Index) car ils sont mieux optimisés (taille de page d'index plus grande).

Les index classiques (appelés Radix ou Binary) ont de meilleurs performances sur des tables de taille relativement modérées et avec des requêtes retournant moins de 20% des lignes de la table.

Les index EVI (Encoded Vector Index, uniquement en SQL) sont à privilégier pour de grandes tables (plusieurs milions de lignes) et pour des requête retournant entre 20% et 60% de lignes.

Index Advisor (V5R4 +)

Une aide à la création d'index recommandés est disponible à partir de la V5R4 accessible de deux manières :

- iSeries Navigator
- Table : QSYS2/SYSIXADV

L'Index Advisor indique les index dont la création est recommandée en fonction de statistiques d'utilisation.

La requête suivante donne la liste des index les plus souvent nécessaires :

```
SELECT substr(TABLE_NAME, 1, 10) table , TABLE_SCHEMA,
substr(KEY_COLUMNS_ADVISED, 1, 20) keys, INDEX_TYPE,
TIMES_ADVISED, ESTIMATED_CREATION_TIME, REASON_ADVISED,
LOGICAL_PAGE_SIZE, MOST_EXPENSIVE_QUERY,
AVERAGE_QUERY_ESTIMATE, TABLE_SIZE, NLSS_TABLE_NAME,
NLSS_TABLE_SCHEMA, MTI_USED, MTI_CREATED, LAST_MTI_USED
FROM QSYS2/SYSIXADV
ORDER BY TIMES_ADVISED DESC
```

Optimisations SQL : TABLES ou PF/LF ?

Le système contrôle la cohérence du contenu des données d'un enregistrement différemment suivant que le fichier a été créé en « natif » (CRTPF/CRTLF) ou par SQL (CREATE TABLE)

- Les données d'un fichier natif sont contrôlées lors des lectures
- Les données d'une table SQL sont contrôlées à l'écriture

Statistiquement, il y a environ 75% de lectures pour 25% d'écritures dans les applications…

Note : De même que les PF et LF peuvent être utilisés en SQL, les Tables SQL peuvent être utilisées en natif dans les programmes …

Sécurité, droits

Sécurité IBM i : un peu de « provoc »

Les spécialistes de la sécurité (auditeurs, etc.) constatent de plus en plus que **les serveurs IBM i sont moins sécurisés que les serveurs Windows !**

La sécurité sur IBM i serait-elle moins bonne que ces autres systèmes ?

Non, bien sûr! Le problème est que :

- les mécanismes de sécurité de l'IBM i (et ses ancêtres AS/400 et S/38) ont toujours été excellents
- donc nous faisons entièrement confiance au système sans nous préoccuper fortement de la sécurité
- La sécurité sur Windows a longtemps laissé à désirer …
- Les administrateurs de serveurs Windows y sont donc particulièrement attentifs et s'en préoccupent fortement....

Sécurité IBM i : le poids du passé

Un autre soucis est que, bien qu'améliorant constamment les mécanismes de sécurité, IBM à le soucis de fournir des versions d'OS garantissant la compatibilité ascendante de ces versions.

Donc les nouveaux mécanismes existent mais ne sont pas toujours activés.

On en arrive sur certains sites à des configurations de sécurité datant du système 38, voire des S/34 et S/36.

Autre élément : d'un système fermé sur lui même, uniquement accessible par des terminaux passifs, l'IBM i est devenu un serveur complètement tourné vers les réseaux et le monde extérieur.

Avec les terminaux passifs, la sécurité par les menus suffisait.

Aujourd'hui il faut revoir la sécurité par la mise en place de droits d'accès sur les objets.

Sécurité IBM i : quelques principes et outils

Un droit spécial à bannir des profils : *ALLOBJ. *ALLOBJ outrepasse les restrictions de droits que l'on peut mettre sur les objets.

2 mécanismes facilitent la vie des administrateurs de la sécurité :

- les profils de groupe (1 groupe principal et jusqu'à 15 secondaires)
- les listes d'autorisations (*AUTL)

Un profil d'administrateur de la sécurité (*SECADM) ne devrait pas avoir le droit *ALLOBJ. (il ne pourra donc pas non plus l'attribuer).

Ne pas oublier qu'un profil est lui même un objet, avec un propriétaire et des droits sur cet objet...

L'IBM i supporte des mots de passe complexes (128 caractères, sensible à la casse, espaces et caractères spéciaux possibles). Le changement se fait pas la SysVal : QPWDLVL

Sécurité IBM i : planification de l'activation des users

Il possible de planifier l'activation et la désactivation de profils utilisateurs à des heures et jours de la semaine spécifiques :

DSPACTSCD : Display Activation Schedule

CHGACTSCDE : Change Activation Scd Entry

```
                    Change Activation Scd Entry (CHGACTSCDE)

Type choices, press Enter.

User profile . . . . . .                    Name
Enable time  . . . . . .                    Time, *NONE
Disable time . . . . . .                    Time, *NONE
Days . . . . . .  . . . .   *ALL            *ALL, *MON, *TUE, *WED..
       • for more values
```

Permet de garantir que des profils ne pourront être utilisés en dehors des heures et jours prévus.

Sécurité IBM i : planification de l'expiration des users

Il est également possible de forcer l'expiration de profils à une date donnée (désactivation ou suppression)

DSPEXPSCD : Display Expiration Schedule

CHGEXPSCDE : Change Expiration Scd Entry

```
                  Change Expiration Scd Entry (CHGEXPSCDE)

Type choices, press Enter.

User profile . . . . . .                    Name
               + for more values
Expiration date  . . . . . .                Date, *NONE
Action . . . . . .   *DISABLE      *DISABLE, *DELETE
```

On s'assure ainsi qu'un utilisateur temporaire (stagiaire, consultant, etc) du système n'y aura plus accès à la fin de sa mission.

Sécurité IBM i : les temps changes...

Les principes établis depuis l'origine du système 38 sont remis en cause avec les possibilités multiples d'accès au système.

Cas classique :

Un User TOTO doit gérer les données des fichiers PARTICLES et PSTOCK à l'aide du programme GSTOCK

La logique voudrait que l'on donne les droits d'utilisation du programme GSTOCK à TOTO et les droits de modifications des données des fichiers PARTICLES et PSTOCK à TOTO.

Or TOTO à besoin pour son travail d'extraire des données de PARTICLES dans EXCEL à l'aide du Plugin (ou d'ODBC, ou du transfert de fichier Client Access)

Le problème est que le profil TOTO permet la modification complète des données de ce fichier, mais également de PSTOCK.

Il peut donc extraire toutes les données, **mais également les mettre à jour !**

Sécurité IBM i : les temps changes...

Avec l'application classique des autorisations d'objet ce problème n'est pas gérable.

Que préconise IBM pour ces cas ? Utiliser l'héritage des droits du propriétaire d'un programme (*OWNER)

- Créer un profil propriétaire des objets : TITIOWN
- Changer le propriétaire des programmes et fichiers en TITIOWN
- changer les programme pour qu'ils s'executent avec les droits de leur propriétaire : CHGPGM PGM(GCLIENTS) USRPRF(*OWNER)
- Donner les droits d'exécution des programmes à TOTO
- Donner le droit de lecture du fichier PARTICLE à TOTO.

TOTO pourra ainsi travailler normalement à l'aide du programme GSTOCK, mais ne pourra qu'extraire les données de PARTICLE.

IFS, QSH, PASE

Qu'est ce que l'IFS ?

L'IFS est le système de fichier de l'IBM i. Ou plus exactement un ensemble de systèmes de fichiers intégrés sous une même racine (/).

IFS signifiant Integrated File Systems.

QSYS est intégré à l'IFS sous le terme QSYS.LIB.

Système arborescent de répertoires (directories) généralement compatible UNIX/LINUX ou Windows.

Les premiers messages de QSYSOPR après un IPL indiquent les systèmes de fichiers « montés » (activés)

```
IPL exécuté à partir de la zone machine ££MACH£B.
Système de fichier Slash (/) monté.
Système de fichier /QOpenSys monté.
Système de fichier /QDLS monté.
Système de fichier /QSYS.LIB monté.
Système de fichier /QOPT monté.
Système de fichier /QFileSvr.400 monté.
Système de fichier /QNTC monté.
Système de fichier /dev/QASP01 monté.
```

Cas de QDLS

QDLS est le plus ancien système de fichier de type DOS sur AS/400.

Créé pour les besoin d'Office Vision/400 (Ttx, Messagerie, Agenda), et également utilisé pour les premiers stockages et échanges de fichiers avec le monde DOS puis Windows.

Limité à des noms de 11 caractères : 8+3 (les 3 derniers étant l'extension, telles que .exe, .txt...)

Bien que « monté » à la racine de l'IFS, ce n'est pas un système de fichier à part entière. Les dossiers et documents sont en réalité sockés dans des objets de la bibliothèque QDOC (objets *FLR et *DOC).

Pour des questions de performances, il est recommandé de ne plus utiliser QDLS.

Mais : les commandes agissant sur QDLS ne sont pas compatibles avec les autres systèmes de fichiers (commandes sur FLR, DOC et DLO)

Gérer l'IFS par des commandes

De nombreuses commandes de l'OS donnent accès aux fichiers et répertoires de l'IFS.

- MD, MKDIR, RD, RMDIR : créer, supprimer un répertoire
- CD, CHDIR : changer le répertoire en cours
- WRKLNK : gérer fichiers et répertoires
- DEL, ERASE, RMVLNK : supprimer un fichier stmf
- RNM, MOV, CPY : Renommer, déplacer ou copier un fichier STMF
- SAV/RST : sauvegarde Restauration
- CHGATR : changer attributs fichier (readonly, hidden ...)
- CHGOWN, CHGAUT : changer propriétaire, droits
- CHKOUT, CHKIN : Verrouiller, libérer un fichier
- Etc …

Sécurité et IFS

Les fichiers et répertoires de l'IFS bénéficient de la combinaison de 3 mécanismes de sécurité :

- les droits OS/400
- les droits de type UNIX (chown, chmod)
- les protections de type DOS (attributs lecture seule, caché …)

NetServer – partage de fichier SMB

L'IBM supporte le partage de fichiers du monde Windows (« Voisinage » ou « Favoris » réseau) connu sous le nom de SMB ou Samba.

Gestion des répertoires partagés soit par System i Navigator, soit par des APIs.

Ce mécanisme permet à des postes de travail ou des serveurs Windows ou Linux d'accéder aux répertoires et fichiers de l'IBM i comme avec n'importe quel autre serveur.

IBM i étant un pur serveur SMB, il n'est pas nécessaire d'installer iSeries Access pour bénéficier de ces partages.

Mais l'inverse est également possible : l'IBM i est un client pour les postes Windows et Linux partageant leurs fichiers !

NetServer – QNTC : voisinage réseau sur IBM i

Le répertoire /QNTC permet de voir les serveurs et postes du réseau qui utilisent SMB.

Si ces postes partages des répertoires, ils seront accessibles de l'IBM i.

WRKLNK '/QNTC/*' :

```
Répertoire . . . . :   /QNTC
Opt   Lien objet              Type      Attribut      Texte
      ARNAUD                  DDIR
      PC1                     DDIR
      PC3                     DDIR
      PC4                     DDIR
      PORT02-7                DDIR
```

Pour avoir accés aux partages d'un poste, il faut que ce dernier ait un compte utilisateur identique au profil utilisateur tentant l'accès, avec mot de passe identique.

Parfois un poste n'est pas visible dans /QNTC. Pour y accèder, il suffit de créer manuellement un répertoire dans QNTC :

MKDIR DIR('/QNTC/SERV123')

/QFileSvr.400 : Partage de l'IFS entre IBM i

L'IBM i dispose de son propre système de partage avec d'autres IBM i.

Le partage s'effectue par le répertoire **/QFileSvr.400**

Pour accéder à un autre système, il faut :

- attribuer un nom à l'@ ip du système cible par CFGTCP option 10
- créer un répertoire portant de nom dans /QFileSvr.400 :

exemple : MKDIR DIR('/QFileSvr.400/AS400N2')

C'est tout. Pour peu que les profils et password soient identiques bien sûr, tout l'IFS de la machine cible est accessible !

```
Répertoire . . . . :   /QFileSvr.400/AS400N2

Opt    Lien objet              Type
       bin                     SYMLNK->DIR
       dev                     DIR
       etc                     DIR
       home                    DIR
.../...
```

Qshell, PASE

Qshell (ou Qsh) et Pase sont des interpréteurs shell unix basés sur une version allégée d'AIX.

Peu de différence entre les 2.

Nécessite les options 30 et 33 de l'OS (57xxSS1)

Appel des terminaux :

- commande QSH ou QSHELL
- CALL QP2TERM pour PASE

Terminaux permettant de passer des commandes UNIX.

La commande QSH permet d'exécuter directement une commande sans passer par le terminal :

```
QSH CMD('mkdir /home/common_romandie')
```

Qshell, PASE : iSeries Tools for Developers

Les outils (commandes Unix) fournis en standard sont peu nombreux.

D'autres outils peuvent être ajouté par la PRPQ 5799PTL (iSeries Tools for Developers).

Contenu de 5799PTL : http://systeminetwork.com/node/60709

pour télécharger :

http://www-01.ibm.com/support/docview.wss?uid=swg21243973

Contient Perl, VNC, Bison, emacs, icc (appel du compilateur C/C++), gzip, OpenSSH, etc.

Qshell, PASE : outils d'AIX

AIX fonctionnant sur Power, les programmes précompilés AIX du domaine libre peuvent être installés dans le PASE.

Le site suivant permet de télécharger une très grande quantité d'outils AIX :

http://bio.gsi.de/DOCS/AIX/aixpdslib.seas.ucla.edu/categories.html

Qshell, PASE : DB2

QSH permet d'appeler la commande AIX : db2.

Cette commande exécute des instructions SQL.

La convention d'appellation est celle de SQL : Library.Table

exemple :

```
QSH CMD('db2 "insert into jeanmichel.ptoto values(''Lausanne 06/2010'')"')
```

QSH étant une commande OS/400 classique, elle permet à l'aide de la commande AIX **db2,** d'**écrire, mettre à jour et supprimer des enregistrements dans des fichiers BdD en CLP** !

IFS : ne pas oublier...

Ne pas oublier que l'IFS contient des répertoires et fichiers qui peuvent être critiques pour le bon fonctionnement du système et des applications.

Donc, comme tout objet de l'AS/400 la sécurité doit être pensée pour l'IFS également :

- **Sauvegardes** (commande SAV)
- **Gestion des droits** (par CHGAUT/CHGOWN ou WRLNK)
- journalisation (STRJRN)
- Audit si besoin

IFS, Qshell, PASE : Conclusion

Avec l'IFS, le language C, les APIs UNIX, QSHELL et PASE, l'IBM i est un vrai système UNIX !

L'AS/400 est validé système UNIX depuis 1993...

Bien que d'autres système UNIX (System V, etc ...) existaient déjà, ce fût le premier système à obtenir cette validation !

pour des raisons financières essentiellement ;)

Divers

Sauvegardes sur unités virtuelles

Les dernières versions de l'OS ont apporté la possibilité d'utiliser des unités optiques (CD) et bandes virtuelles permettant de

- charger des images .iso (ou .bin) de CD/DVD dans l'IFS pour les traiter par les commandes habituelles au travers d'une unité virtuelle
- Ecrite et Lire à l'aide d'unité de bandes virtuelles dans des fichiers de l'IFS.

Avantages :

- Rapidité du traitement sur disque
- simplicité de traitement (pas de manipulation de supports en cas de dépassement de capacité bande, plusieurs CD à lire ...)
- possibilité de transférer un fichier contenant une sauvegarde via le réseau (FTP...) sur un autre serveur
- avantage par rapport à un SAVF : plusieurs bibliothèques à la fois

Création d'une unité de bande virtuelle :

Création et activation d'une unité virtuelle :

- `CRTDEVTAP DEVD(`**`VRTTAP1`**`) RSRCNAME(*`**`VRT`**`) TEXT('Unité bande vrt')`
- `VRYCFG CFGOBJ(VRTTAP1) CFGTYPE(*DEV) STATUS(*ON)`

*Création d'un *IMGCLG (sorte de médiathèque) :*

- `CRTIMGCLG IMGCLG(`**`IMGTAP1`**`) DIR('`**`/vrttap`**`') TYPE(*TAP) CRTDIR(*YES)`

*Ajout d'un « support » dans l'*IMGCLG et chargement :*

- `ADDIMGCLGE IMGCLG(`**`IMGTAP1`**`) FROMFILE(*NEW) TOFILE(`**`Tapefile`**`)`
- `LODIMGCLG IMGCLG(`**`IMGTAP1`**`) DEV(`**`VRTTAP1`**`) OPTION(*LOAD) WRTPTC(*NONE)`

Utilisation d'une unité de bande virtuelle :

Une fois crée, une unité de bande virtuelle s'utilise de la même manière qu'une unité classique.

- `INZTAP  DEV(VRTTAP1) NEWVOL(SAVE1) CHECK(*NO)`
- `SAVLIB  LIB(QGPL)  DEV(VRTTAP1)`
- `DSPTAP  DEV(VRTTAP1) DATA(*SAVRST)`
- …

Le résultat de la sauvegarde est un fichier dan l'IFS. Dans notre exemple, c'est : /vrttap/Tapefile

API QUILNGTX : Une fenêtre de texte

Cette api permet très simplement d'afficher une fenêtre de texte.
Utilisation simple dans un programme RPG ou CLP

```
CALL PGM(QUILNGTX) PARM('Bienvenue à Lausanne !' X'00000016' '' ''
 X'0000000000000000')
```

affichera :

API : User Application

3 Apis sont apparues en V5R3 et permettent de créer des Informations d'application intimement associées à un User Profile.

Cela permet par exemple d'associer des préférences utilisateurs (OUTQ, ordre de tri, dernière recherche effectuée, etc ...)

Chaque information ajoutée à un identifiant (max 200 car) et les données elles mêmes (max 1700 char).

L'association directe au profil utilisateur fait que ces informations sont sauvegardées et restaurées en même temps que le profil

- Update User App. Information (QsyUpdateUserApplicationInfo)
- Retrieve User App. Information (QsyRetrieveUserApplicationInfo)
- Remove User App. Information (QsyRemoveUserApplicationInfo)

En savoir plus …

EXPERIA Europe

Forums en français : http://forum.commonfr.org

Forums en Anglais : http://www.code400.com/forum

RPG Café d'IBM :

http://www-949.ibm.com/software/rational/cafe/community/rpg

EXPERIA Europe
Les Jardins d'Epione
4, rue L. BERIDOT
38500 Voiron

Tel : 04 76 67 07 70
Fax : 04 76 05 40 55

eMail : info@experia.com

http://www.experia.com