COBOL

COmmon Business-Oriented Language

Présentation

- Historique
  - 1960: première définition du langage par CODASYL.
  - ANS 68, ANS 74, ANS 85 par ANSI.
  - ANS 68: résout les problèmes d'incompatibilités entre les différents versions.
  - ANS 74: introduction du verbe CALL pour appeler des programmes externes.
  - ANS 85: programmation structurée (END IF. , END READ.) et sous programmes internes
  - ISO 2002: introduction des concepts de l'orienté, unicode et XML dans le langage COBOL.
  - ISO 2014: amélioration du support multi-plateforme.
- Caractéristiques
  - Auto documenté: les instructions (verbes) du langage sont inspirés de la langue anglaise.
  - Stabilité: depuis sa création seulement 5 standards ont été réalisés avec une forte compatibilité descendante.
  - Supporté par tous les systèmes existants: Windows*, Unix* , IBM VMS, zOS, zVSE, HP-UX, GCOS , DPX de Bull, DEC PDP-11/70, VAX, Univac, SuperNova, General NOVA, ECLIPSE MV de Data General, EX33 de Hitachi

Structure d’un programme Cobol

- Un programme Cobol est constitué de quatre divisions
- Une division est composée de sections, une section est constituée de paragraphes contenant des phrases (instructions), une phrase doit se terminer par un point.
- IDENTIFICATION DIVISION: identification du programme (nom du programme, développeur)
- ENVIRONMENT DIVISION: association des noms des fichiers utilisés dans le programme (noms symboliques) aux fichiers stockés dans le système.
- DATA DIVISION: description de la structure physique des enregistrements et spécification de la sortie du programme, les données peuvent provenir des fichiers, tables de bases de données, ou structures internes au programme .
- PROCEDURE DIVISION: instructions du programme.

Exemple d’un programme cobol

```
      *A 1 B  2   3    4
       identification division.
       program-id. Exemple1.
       author. Cobol1.
      *   Premier programme cobol
       environment division.

       data division.

       procedure division.
           display 'Hello World'.

       stop run.
```

- Program-id: identification du programme.
- Author: nom de l’auteur du code source.
- Les colonnes 1 à 6 sont réservées pour un numéro de séquence.
- La colonne 7 est une colonne spéciale, si elle contient un * , alors la ligne est un commentaire.
- Les noms divisions , des sections et des paragraphes; doivent commencer dans la zone 8-11 (de la colonne 8 à la colonne 11)

La division Identification

```
identification division.
program-id. Exemple1.
author. Cobol1.
date-written. 6 juillet.
installation. septembre.
date-compiled. premier octobre.
```

- Program-id peut être remplacé par class-id dans le cas dans du cobol orienté objet.
- Program-id:
  - D’autres programmes peuvent appeler ce programme par son id

La division Environnement

```
environment division.
configuration section.
source-computer. zos.
object-computer. zos.
input-output section.
file-control.
    select dataIn assign to inFile.
    Select dataOut assign to outFile.
```

- Peut contenir deux sections:
  - Configuration section: identifie l'environnement actuel, machine de compilation, machine d'exécution, symbole monétaire…
  - Input-output section: identifie et décrit les caractéristiques des fichiers utilisés par le programme

La division Data

```
data division.
file section.
fd dataIn.
01 dataInEnreg pic X(80).
fd dataOut.
01 dataOutEnreg pic X(80).
working-storage section.
01 champ1 pic x(20).
01 compteur pic 99.
linkage section.
01 ls1 pic x(10).
```

- La division peut contenir les sections:
  - File Section
    - FD (File Definition): déclare que les fichiers contiennent des enregistrement de 80 caractères.
  - Working-Storage Section: allocation statique des variables, champ1 est une variable alphanumérique de 20 octets, compteur un entier de deux chiffres.
  - Local-Storage section: l'espace alloué est libéré à chaque retour vers le programme appelant.
  - Linkage section: variables déclarées dans section working-storage du programme appelant (elle sert à passer des valeurs entre le programme appelant et le programme appelé)

La division Procedure

```
procedure division using ls1.
000-Main Section.
000-Begin.
    open input dataIn
    open output dataOut
    perform 010-lire-ecrire
    close dataIn dataOut
    stop 'Tapez la touche R pour Terminer'
    stop run.
010-lire-ecrire section.
010-begin.
    read dataIn
    move dataInEnreg to dataOutEnreg
    write dataOutEnreg
    display dataOutEnreg.
    stop run.
```

- Toutes les divisions peuvent contenir des sections et des paragraphes.
- Les noms des sections et des paragraphes dans les 3 premières sections sont prédéfinis, alors que pour la 4^{e} division ils sont définis par l'utilisateur.
- Les noms des divisions et des sections doivent se terminer par un point.
- Les instructions des 3 premières divisions doivent se terminer par un point.
- Dans la division « procedure » la dernière instruction d'un paragraphe doit se terminer par un point.

Identificateurs

- Un identificateur doit avoir entre 1 et 30 caractères (alphanumérique ou le symbole -), il doit contenir au moins une lettre et ne doit pas commencer ou se terminer par -.

Hiérarchisation des données

```
01  EMPLOYE-RECORD.
    03  EMPLOYE-NOM.
        05  EMPLOYE-TITRE          PIC X(3).
        05  EMPLOYE-INITIALES       PIC X(4).
        05  EMPLOYE-SURNOM         PIC X(30).
    05  EMPLOYE-SEXE              PIC X.
    05  EMPLOYE-ADRESSE.
        07  EMPLOYE-LIGNE-ADRESSE   PIC X(30) OCCURS 4.
    05  EMPLOYE-CODE-POSTAL        PIC X(8).
    05  EMPLOYE-SALAIRE             PIC 9(5)V99.
```

- Lors de la déclaration d'une variable dans la section data nous devons aussi déclarer le niveau d'hiérarchie de la variable le premier niveau est désigné par 01 et doit commencer à partir de la colonne 8, les autres niveaux doivent commencer à partir de la colonne 12.

La clause PICTURE

- La clause PICTURE (PIC) permet de définir les types de données élémentaires
  - X: caractères alphanumériques
  - A: alphabétique.
  - 9: un chiffre
  - PIC X(4) est identique à PIC XXXX
  - Un champ numérique peut contenir un maximum de 31 chiffres.
  - V représente la virgule
  - PIC 9(5)V9(2) est identique à PIC 99999V99.
  - S: désinge un nombre signé, exemple: PIC S9(5)V99.
- Les littéraux
  - Un litéral numérique peut contenir un maximum de 268 434 912 caractères.
  - Un littéral numérique peut contenir au maximum 31 chiffres.
  - Exemples 03 C1 PIC X(6) value "COBOL".

    03 NUM PIC 9(6) value 76676.

Les constantes figuratives

SPACE, SPACES	`NOM PIC X(20) VALUE SPACE.`
ZERO, ZEROS, ZEROES	`var1 PIC 9(4) VALUE ZERO.`
HIGH-VALUE, HIGH-VALUES	constante formée de 1 binaires "partout"
LOW-VALUE, LOW-VALUES	constante formée de 0 binaires "partout"

- Pour initialiser tous les champs d'un enregistrement:

```
INITIALIZE enr-emp.
```

Les verbes

- Chaque instruction de la division « procedure » doit commencer par un verbe.
- Exemples de verbes:
  - MOVE
  - DISPLAY
  - ACCEPT
  - STOP
  - IF
  - Perform
  - Add
  - Mulitply
- Le verbe move copie un élément de données dans une ou plusieurs variables : move v1 to v2 v3.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.   exemple3.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  chaine    PIC X(20) VALUE SPACES.
01  nom       PIC X(20) VALUE SPACES.
PROCEDURE DIVISION.
S1 SECTION.
p1.
    MOVE 'Bonjour' TO chaine
    DISPLAY chaine
    DISPLAY 'Entrez votre:'
    ACCEPT nom
    DISPLAY 'Votre nom: ' nom
    MOVE 'FIN' TO chaine
    DISPLAY chaine
    STOP 'TAPEZ UNE TOUCHE POUR TERMINER'
    STOP RUN.
```

Le verbe IF

```
IF EMPLOYE-SALAIRE  < 40000
DISPLAY 'salaire inférieur à 40000'
END-IF

Ou bien
IF EMPLOYE-SALAIRE  < 40000
       DISPLAY 'salaire inférieur à 40000'
ELSE
        DISPLAY 'salaire supérieur à 40000'
END-IF
```

- Opérateurs de comparaison
  - =, <,>,not =, <>, unequal, not equal to
- Signes:is negative, is positive, is zero
- Is alphabetic, is numeric
- Conditions composées: AND , OR
- Poour quitter un bloc IF : next sentence

EVALUATE

```
EVALUATE FONCTION
WHEN 'A' THRU 'D'
PERFORM ADMIN
WHEN 'E'
PERFORM ENSEGEIGNANT
WHEN 'G'
PERFORM GERANT
WHEN 'M'
PERFORM MEDECIN
WHEN OTHER
PERFORM INVALID
END-EVALUATE
```

Le verbe PERFORM

- Le verbe perform permet d'exécuter un sous programme:

  Perform sous-prog (sous-prog est le nom d'un paragraphe ou d'une section)

- `Perform` sous-prog `until` condition: exécution du programme tant que « condition » est vraie.
- `Perform` sous-prg `with test after until` condition : le sous programme est execute au moins une fois.

Les verbes ADD DIVIDE MULTIPLY

- `Add N1 to N2`
- `Add N1 N2 1000 to N3 N4`
- `Substract N1 from N2`
- `Substract N1 N2 from N3 GIVING N4`
- `Multiply N1 by N2 (N2=N1*N2)`
- `MULTIPLY 1.01 BY N1 N2`
- `MULTIPLY N1 BY 1.01 GIVING N1 N2`
- `Multiply N1 by N2 giving N3`
- `Divide N1 by B2 giving N3`

```
COMPUTE WS-INVOICE-AMT = WS-ITEM-PRICE * WS-QUANTITY
     ON SIZE ERROR
     PERFORM ERROR-ACTIONS
     NOT ON SIZE ERROR
     PERFORM SET-UP-INVOICE
END-COMPUTE
```

La clause usage

- La clause usage détermine le format interne de stockage des données
- Valeurs possibles
  - usage display (Valeur par défaut)
    Compteur pic 9(4).
    l'allocation est réalisée par octet, chaque chiffre occupe un octet.
  - Usage NATIONAL
  - Format binaire
    Compteur pic 9(4) comp. (ou usage computational. ou binary.)
    l'allocation est effectuée par bit, par exemple 9999 occupe 14 bits (2 octets)

  - Packed decimal format : pic 9(4) comp-3.

Fichiers séquentiels

Le cobol support 3 types de fichiers

- Séquentiel
  - Relatif
  - Indexé

Etapes d'utilisation d'un fichier séquentiel

1. Déclaration dans la division « Environment »
2. Description des enregistrements du fichier dans la division « Data ».

3. Utilisation du fichier dans la division « Procedure »

1. Ouverture

Syntaxe Open mode_accès nomFichier

il existe 3 modes d'accès pour les fichiers séquentiels:

- INPUT: ouverture en mode lecture, le fichier doit exister.
- OUTPUT: ouverture en mode écriture, le fichier sera créé, s'il existe il sera écrasé.
- EXTEND: le fichier est ouvert en mode append (ajout à la fin du fichier)

2. Manipulation

Verbes possibles: READ, WRITE, CLOSE

- Le verbe READ lit l'enregistrement suivant et le stocke dans la variable définie par FD, Exemple :

  READ INFILE

- Le verbe WRITE enregistre le contenu de l'enregistrement décrit par FD dans le fichier correspondant.

  WRITE outFileEnreg.

Exemple 1: Ecriture séquentielle dans un fichier

```
program-id. fiche1 as "fiche1".

    environment division.
    input-output section.
    file-control.
    select outF assign to 'd:/temp/fichier.dat'.
    configuration section.

    data division.
    file section.
    fd outF.
    01 enr.
        03 nom PIC X(20).
        03 montant PIC 9(5).
        03 date_operation.
            05 jour PIC 99.
            05 mois PIC 99.
            05 annee PIC 9999.

    working-storage section.
    01 ws-enr.
        03 nom PIC X(20).
        03 montant PIC 9(5).
        03 date_operation.
            05 jour PIC 99.
            05 mois PIC 99.
            05 annee PIC 9999.

procedure division.
debut.
    display 'Hello:'
    open extend outF

    display 'Nom:'
    accept nom in ws-enr
    display 'Montant:'
    accept montant in ws-enr
    display 'Jour:'
    accept jour in date_operation in ws-enr.
    display 'mois:'
    accept mois in date_operation in ws-enr.
    display 'Année:'
    accept annee in date_operation in ws-enr.
    write enr from ws-enr

    close outF.
    display 'FIN!'


end program fiche1.
```

Exemple 2: Lecture séquentielle

```
 program-id. fichier_seq1 as "fichier_seq1".

       environment division.
       input-output section.
       File-Control.
      * Le fichier en lecture doit exister
      *le fichier en écriture s'il n'existe, il sera créé, par
      *défaut s'il existe il sera écrasé.
      *organization sequential. par défaut
       select inF assign 'd:/temp/fichier1.dat' organization sequential.
       select outF assign 'd:/temp/fichier2.dat'.
       configuration section.

       data division.
       file section.
       fd inF.
       01 enrInF.
           03 enrType PIC X.
           03 nom PIC X(20).
           03 dob  PIC 9(8).
       fd outF.
       01 enrOutF.
           03 nom PIC X(20).
           03 dob  PIC 9(8).


       working-storage section.
       01 WS-EOF PIC 9.

       procedure division.
       programme section.
       debut.
           open input inF
           open output outF
           read inF
      *    L'enregsitrement suivant est stocké dans enrInF.
      *    At End est appelé s'il n'existe plus d'enregistrement suivant
      * READ INFILE INTO WS-INREC:
      *     identique à: READ INFILE
      *     MOVE INREC TO WS-INREC
      * WRITE OUTREC FROM WS-OUTREC.
      *      identique à
      *    MOVE WS-OUTREC TO OUTREC
      *    WRITE OUTREC
           READ inF
           AT END
           MOVE 1 TO WS-EOF
           NOT AT END
           PERFORM GOT-A-RECORD
           END-READ
           goback.
       GOT-A-RECORD.


       end program fichier_seq1.
```

Utilisation des tableaux

Déclaration d'un tableau

```
tableau1 pic 99 occurs 10.
01 n   pic 99.
```

Initiliastion

```
perform varying n from 1 by 1 until n > 10
          accept tableau1(n)
          end-perform
```

Lecture

```
perform varying n from 1 by 1 until n > 10
          display tableau1(n)
          end-perform
```

La taille du tableau peut déterminée au moment de l'exécution

```
01 nb pic 999.
       01 tableau3 PIC X OCCURS 0 TO 100 DEPENDING ON nb.
```

Tableaux multidimensionnelles

- **Déclaration**

```
01 donnees.
        03 tableau2 OCCURS 10.
             05 v1 PIC X(20).
             05 tableau21 OCCURS 20.
                  07 v2 PIC X(20).
                  07 tableau212 PIC 9(5)V99 OCCURS 20.
```

Utilisation

```
accept v1(1)
             accept v2(1,1)
             accept tableau212(1,1,1)
```

Mise en forme de l'affichage

Z remplace les zéros à gauche par des espaces

Exemple:

```
01 val1 PIC 9(6)V99.
01 val2 PIC Z(5)9.99.
move 4532.43 to val1
           move val1 to val2
           display val1
                   val2
```

Résultat affiché :00453243 4532.43

Remarques

Val2 ne peut pas être utilisé dans les calculs, il est considéré comme une chaîne de caractères

Z peut être remplacé par *, *(5)9.99 aurait donné: **4532.43

Utilisation du symbole monétaire et du séparateur de millier:

```
01 val1 PIC 9(8)v99.
01 val2 PIC $$$,$$$,$$9.99.
```

Les caractères d'insertion

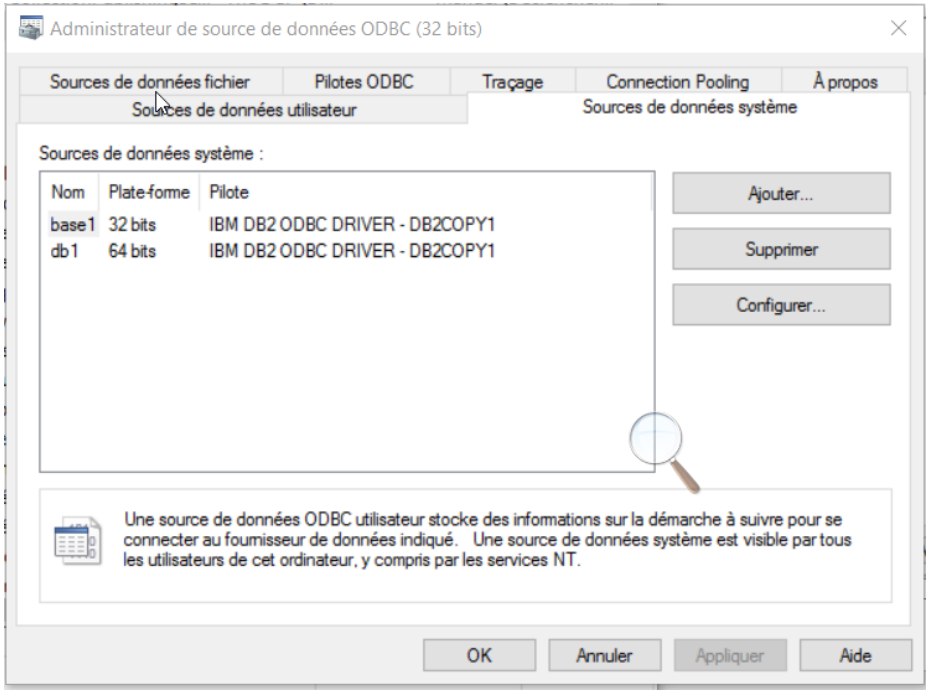
/ (date) , : (heure), B (espace), 0

```
01 dat1 PIC 9999/99/99.
01 cc PIC X(4)BX(4)BX(4)BX(4).
01 WS-SOL PIC 999000.
```

Connexion à une base de données DB2 en utilisant une source ODBC

- Sous une

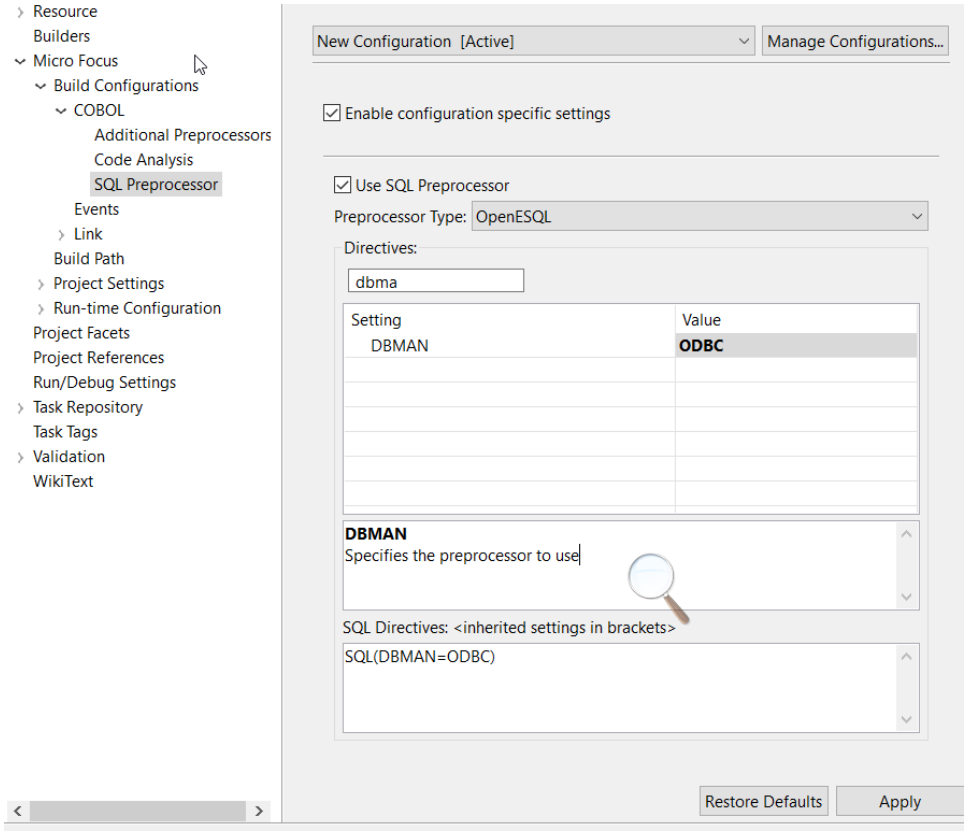
Créer une nouvelle source de données système



1. Dans le panneau de configuration, sélectionner le composant « Configurer les sources de données système ».
2. Dans l'onglet « Source de données système », cliquer sur « Ajouter »
3. Sélectionner le driver « IBM DB2 ODBC DRIVER »
4. Sélectionner la base de données « sample» et donner un nom à la source de données système.

Configurer le projet

- Définir la valeur de la directive dbman à « ODBC »



Connexion à la base de données

Le cobol supporte le langage ESQL (Embedded SQL) qui est un sous ensemble du langage SQL

Les instructions sql doivent être dans un bloc EXEC SQL …. END EXEC.

Connexion à la source de données

dans la procédure division

```
exec sql connect to ‘nom DSN' end-exec.
```

Exécution des instructions SQL

- Instruction SQL (DDL ou DML)

```
exec sql
      select deptname into :deptname
      from department where deptno='D01'
end-exec
display deptname
```

On pourrait aussi sélectionner tous les champs dans la variable dep

```
select * into :dep
```

Variable déclarée dans la section working-storage.

```
01 DEP.
      03 DEPTNO PIC X(4).
      03 deptname pic X(60).
      03 mgrno pic x(20).
      03 ADMRDDEPT pic x(4).
      03 LOCATION pic X(20).
```

SQLCA

SQL Communication Area est une structure contenant un ensemble de variable, SQLSTATE est une variable de la structure SQLCA (de type PIC X(5)) et elle contient le code du statut de la dernière instruction SQL exécutée.

- Liste des valeurs possibles de SQLSTATE

  https://www-01.ibm.com/support/knowledgecenter/ssw_ibm_i_71/rzala/rzalaccl.htm
- Principales Valeurs:
  - 00000: exécution sans erreur.
  - 02000: la positon du curseur est après la dernière ligne.

Pour importer la structure SQLCA dans un programme, il faut utiliser le bloc exec sql INCLUDE dans la division data (working-storage section)

```
EXEC SQL INCLUDE SQLCA END-EXEC.
```

Utilisation d'un curseur

Déclaration

```
EXEC SQL
DECLARE crDEP CURSOR FOR SELECT
            * FROM DEPARTMENT
END-EXEC.
```

Ouverture du curseur

```
exec sql
 open crDEP
end-exec
```

Parcours des enregistrements

```
perform until SQLSTATE >= '02000'
                exec sql
                    fetch crDEP into :deptno,:deptname
                end-exec
                if SQLSTATE < "02000"
                display deptno " " deptname
                end-if
end-perform
```