PLAN – Package - Collection

I - La préparation pour exécution

Avant de pouvoir utiliser un programme, il faut transformer le programme source en :
- code exécutable par MVS pour la partie « langage hôte » (load-module)
- code exécutable par DB2 pour la partie « SQL ».

Pour cela, de trois à cinq étapes sont nécessaires :

- la précompilation,
- la compilation
- le link-edit,
- le bind package.
- le bind plan

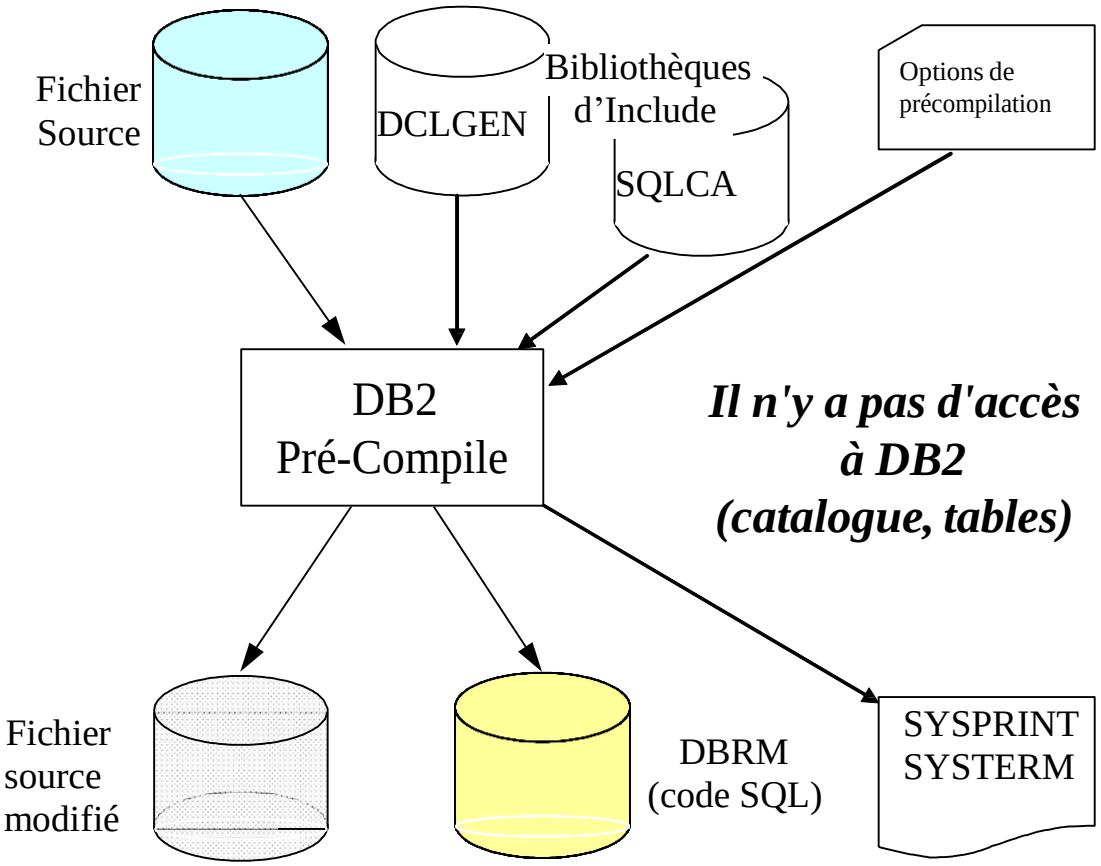
Avec le compilateur Enterprise COBOL for z/OS précompilation et compilation sont faites dans la même étape (dans le même step si on utilise un JCL en batch)

Avec certaines anciennes méthodes de gestion on n'utilisait pas de package (donc pas de bind package).

1 - Etape de précompilation

Le précompilateur DB2 :

- Insère les membres de librairie demandés par EXEC SQL INCLUDE
  - Résultats de DCLGEN
  - Insère la SQLCA
- Vérifie les ordres SQL et les définitions des Host-Variables
- Vérifie la syntaxe SQL
- Traduit les ordres SQL en langage hôte (ici en Cobol) pour la compilation. Une instruction EXEC SQL est remplacée par un appel au module DSNHLI (par un CALL).
- Crée un DataBase Request Module (DBRM) contenant les ordres SQL. Ce DBRM est stocké dans une librairie appelée DBRMLIB.



Le fichier source modifié et le module DBRM se séparent ici.

Evidemment, il n'y a pas génération d'un fichier de sortie : précompilation SQL et compilation COBOL se font dans le même step (« une seule passe »).

L'appel au module DSNHLI se fait avec les paramètres suivants :

1) Nom du programme (= nom du DBRM),
2) VERSION du programme (voir explications plus bas).
3) Un identifiant unique généré par le précompilateur à partir d'un timestamp

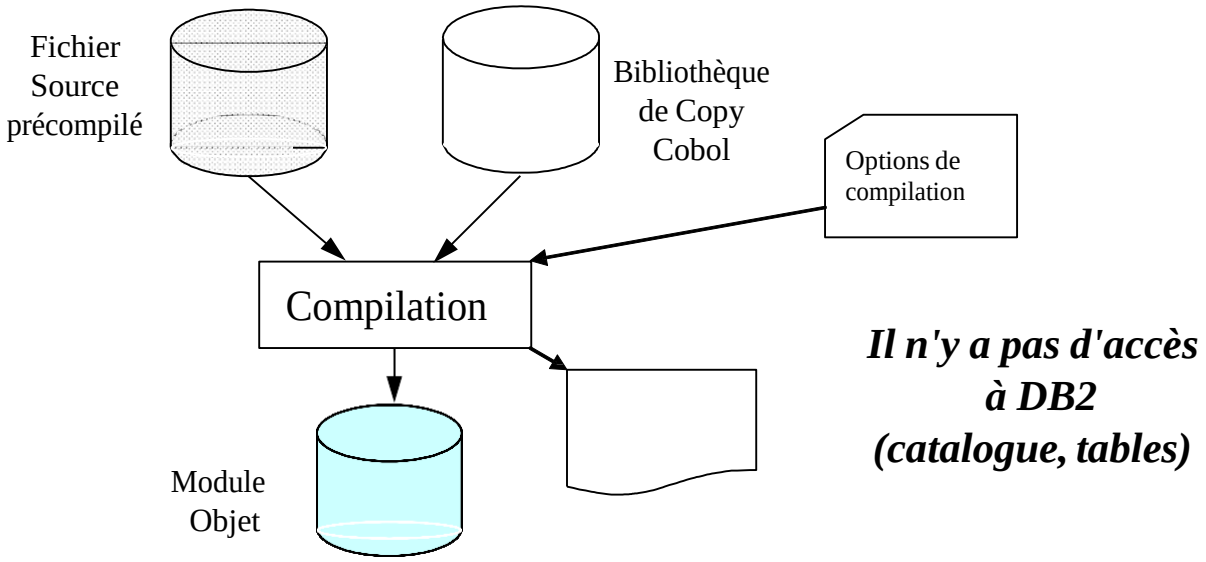
4) Le numéro d'instruction SQL (numéro de la ligne où débute l'instruction, selon la numérotation du source modifié en sortie du précompilateur),
5) Les adresses des Host-Variables,
6) L'adresse de la SQLCA.

les étapes de précompilation SQL et de compilation COBOL sont faites dans le même step (en une seule « passe »)
Le traitement des instructions SQL dans le source COBOL par le coprocesseur SQL associé au compilateur COBOL est fait exactement avec les mêmes principes que le traitement par le précompilateur DB2.

N.B : affichage du contenu du DBRM :
A partir de DB2 Version 8, le DBRM est codé en UNICODE UTF8.
Son contenu n'est donc pas lisible si on l'ouvre sous ISPF en EDIT ou VIEW.
Pour afficher son contenu, il faut l'ouvrir avec le BROWSE ISPF, puis exécuter la commande primaire DISPLAY UTF8 du Browse. Pour revenir à l'affichage normal des caractères codés en EBCDIC, utiliser la commande DISPLAY RESET.

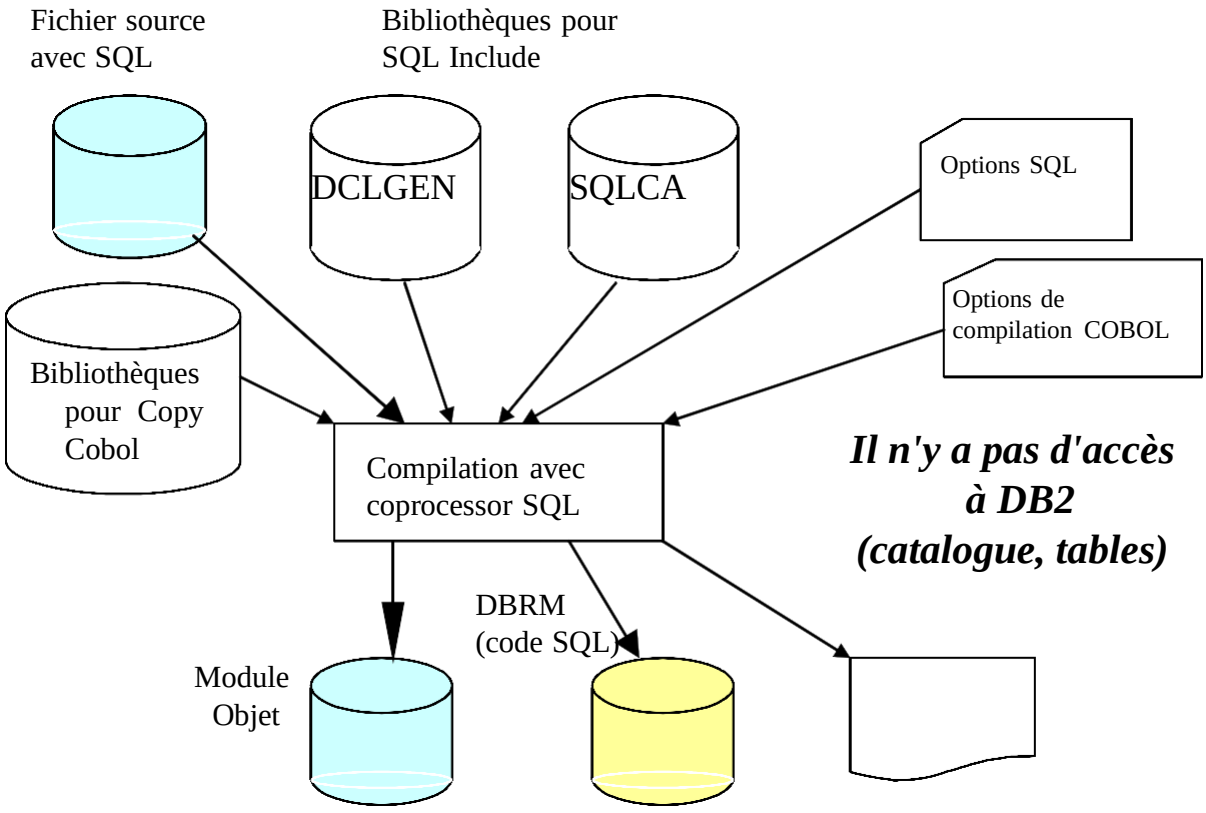
2 - Etape de compilation

Compile le source modifié par le précompilateur pour générer un module objet.



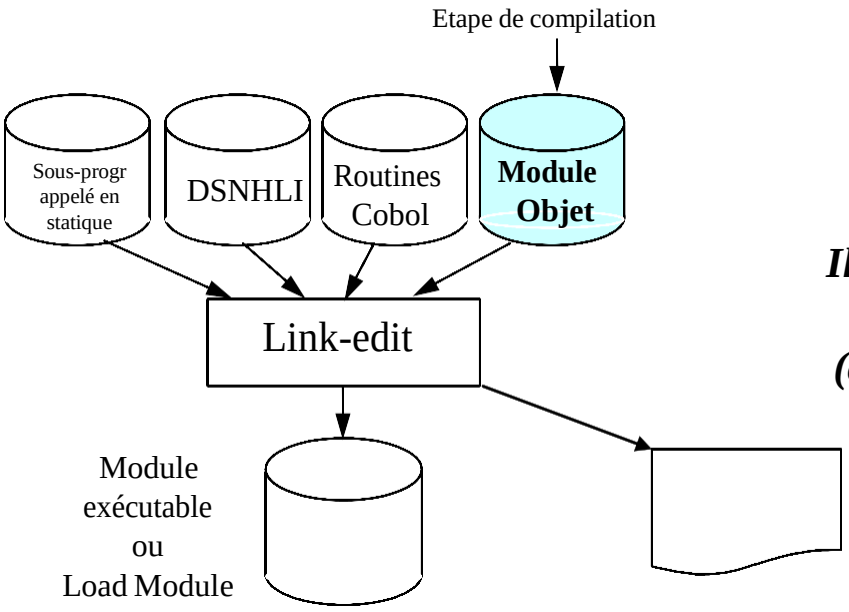
Compilation avec SQL

Si on utilise Enterprise COBOL for z/OS avec SQL, les étapes de précompilation et de compilation sont faites dans le même step batch (une seule « passe »).



3 - Etape de link-edit

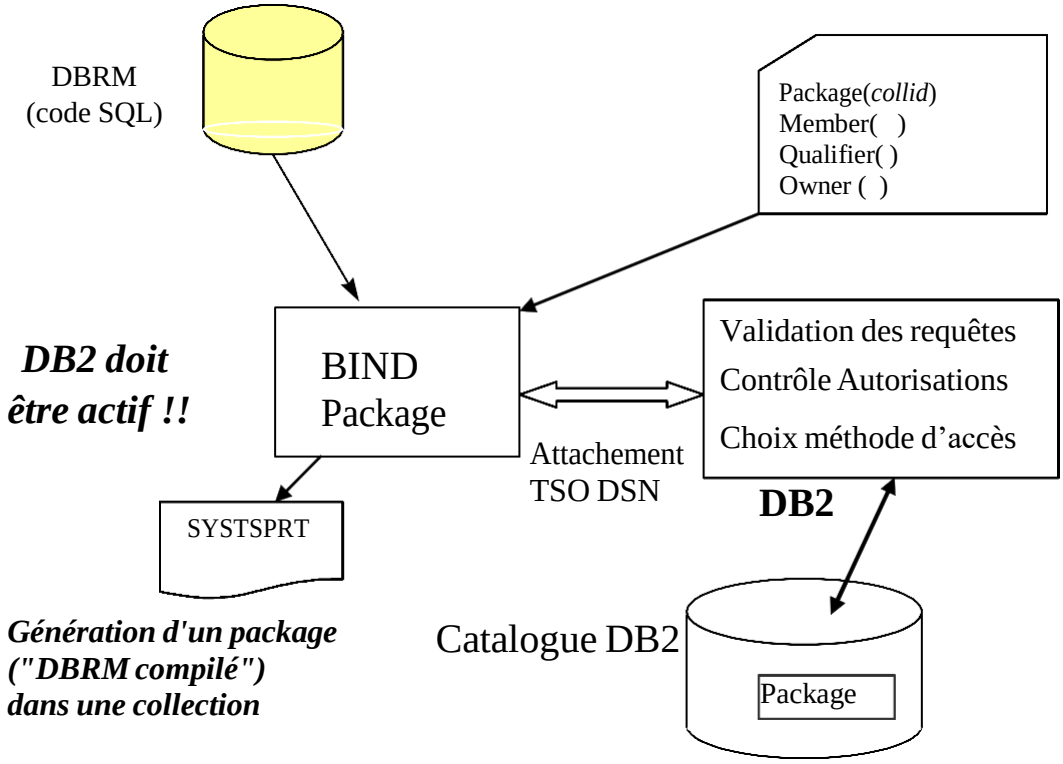
- Link-Edit de différents modules objets en un Load Module (par exemple pour les modules appelés par call statique (si le programme ne fait que des call dynamiques pas d'autre modules à insérer).



Il n'y a pas d'accès à DB2 (catalogue, tables)

4 - Etape de BIND Package

Le Bind Package est en quelque sorte la « compilation » du code SQL contenu dans un DBRM, le résultat de cette compilation est un « package ».



Dans l'étape de BIND PACKAGE, DB2 doit être actif pour accéder aux tables du catalogue DB2 (database DSNDB06) et pour mettre à jour la Directory DB2 (Database DSNDB01).

On fait une commande BIND PACKAGE pour chaque DBRM.

L'option ACTION(REPLACE) permet de demander le remplacement du package s'il existe déjà. On ne peut pas faire le BIND PACKAGE MEMBER(xxxxxx) pendant que le module xxxxxx est en cours d'exécution.

N.B. : Cette étape de Bind package est parfois remplacée par BIND PLAN. Avec un BIND PLAN le résultat de la compilation d'un ou plusieurs DBRM est envoyé directement dans une entité appelée « Plan »

La commande DB2 « BIND PACKAGE » effectue les contrôles et traitements suivants :

- Valide le code utilisateur :
  paramètre "OWNER" de la commande BIND PACKAGE
  ou Userid TSO si Owner n'est pas fourni.

- Valide les requêtes SQL contenues dans un DBRM :
  analyse des objets référencés dans les requêtes tables, vues, colonnes ...
  ces objets sont-ils définis dans le catalogue DB2 (tables SYSIBM.SYSTABLES, SYSIBM.SYSCOLUMNS, SYSIBM.SYSVIEWS, …) ?

- Vérifie les autorisations :
  L'utilisateur a-t'il le privilège requis pour accéder aux objets référencés et pour le type d'accès envisagé (lecture, update, ….) ?
  N.B. : si les noms de tables ne sont pas qualifiés dans le source COBOL, DB2 utilise la valeur du paramètre "QUALIFIER" ("QUAL" en abrégé) fourni à l'appel de la commande BIND pour qualifier les noms de tables, ou à défaut le "Owner" ou le Userid.

- Pour chaque ordre SQL, détermine une méthode d'accès aux données qui sera mise en oeuvre à chaque exécution de l'ordre SQL:
  DB2 choisit une méthode d'accès censée être la plus efficace (la moins coûteuse en I/O et CPU).

Le résultat logique de la commande BIND PACKAGE est une entité appelée PACKAGE, il y a un Package pour chaque DBRM (donc un package pour chaque source compilé).

Le nom du package, « NAME », est égal au nom de membre du DBRM : 8 caractères maximum.

La commande BIND PACKAGE :

- met à jour des tables du catalogue pour conserver les informations en entrée de la commande (tables SYSIBM.SYSPACKAGE, SYSIBM.SYSPACKSTMT, …)

Si l'option **EXPLAIN(YES)** est spécifiée, pour chaque requête SQL contenue dans le DBRM, la méthode d'accès retenue est traduite en informations compréhensibles écrites dans une table **Plan_Table** spécifique au « owner » du Bind.

Privilèges sur le PACKAGE : Il peut être nécessaire, sur certains sites (suivant la valeur du owner fournie à la commande bind, …), de donner des privilèges sur le package aux autres utilisateurs (commande SQL GRANT) de manière à permettre à ces autres utilisateurs d'exécuter le package ou de faire des modifications sur cet objet : BIND PACKAGE après une modification du module COBOL.

5 – Collections

Les sources et les DBRM sont des membres de librairie
(librairie = organisation de stockage propre à MVS : PDS).
Une librairie est un regroupement logique de membres
(un membre = un programme source ou un membre = un DBRM).

De même une collection est un regroupement logique de packages.

La syntaxe de la commande BIND PACKAGE.ressemble à ceci :

```
BIND PACKAGE(nom_de_collection)
MEMBER(nom_du_dbrm)
```

Le nom de la librairie qui contient le DBRM peut être spécifié par l'option LIBRARY (si le bind package est exécuté en batch l'option LIBRARY peut être remplacée par une carte DD du JCL : //DBRMLIB DD DSN=…).

La clé permettant d'identifier de manière unique un package du point de vue de DB2, est constituée de :

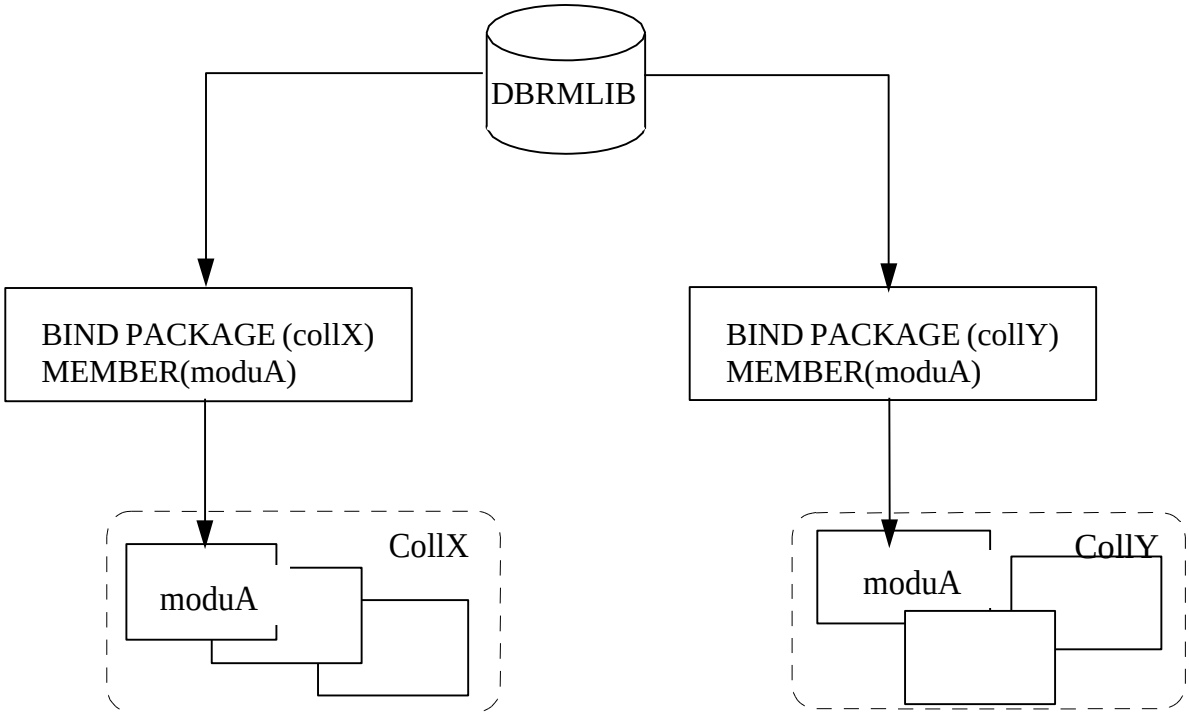
- le nom de la collection « COLLID »
- le nom du package « NAME », qui est égal au nom de membre du DBRM en entrée de la commande BIND PACKAGE (paramètre MEMBER)
- la version du package, qui peut être vide

Si l'option VERSION est fournie en paramètre de la précompilation DB2 (ou en paramètre SQL du coprocesseur SQL associé au compilateur Enterprise COPBOL for z/OS), la valeur fournie dans ce paramètre est utilisée dans la colonne VERSION pour identifier les packages. On peut ainsi avoir plusieurs packages de même nom dans la même collection.

Les collections permettent de regrouper les packages en ensembles logiques.
Le même programme peut se trouver dans plusieurs librairies.
De même **on peut trouver des packages de même nom dans plusieurs collections**.

Par exemple, on pourra gérer deux collections :

- une pour l'environnement de développement :
  nom de la collection ou « COLLID » = 'TEST',
- une pour l'environnement de production : COLLID = 'PROD'.



Les deux packages ci-dessus sont identifiés de la manière suivante :
CollX.moduA et CollY.moduA.

Il peut être utile de faire plusieurs Bind package à partir du même DBRM en envoyant les résultats dans des collections différentes.
Par exemple si plusieurs personnes font des tests nécessitant le même programme (module commun) mais que chacune des personnes dispose de son jeu de tables propre, on fera un bind package avec un paramètre qualifier adéquat pour que les ordres SQL accèdent les bonnes tables au moment de l'exécution.

Exemple : si on a une table TEST1.EMP et une table TEST2.EMP dans le même système DB2, et un programme PGM1 qui accède à la table EMP, on fera :

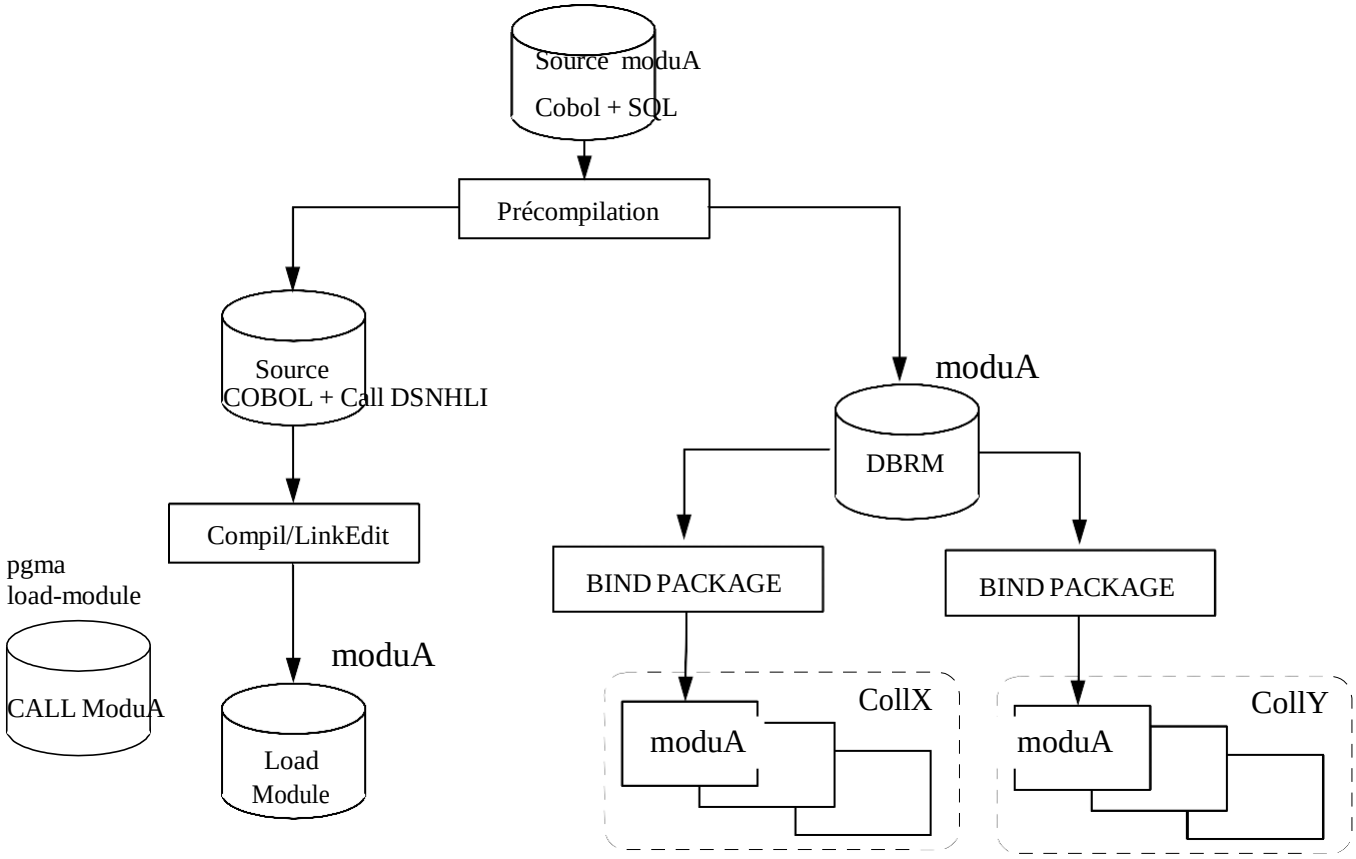
```
BIND PACKAGE(CTEST1) MEMBER(PGM1) QUALIFIER(TEST1)
BIND PACKAGE(CTEST2) MEMBER(PGM1) QUALIFIER(TEST2)
```

Ainsi le user TEST1 lancera l'exécution du programme PGM1 avec le package CTEST1.PGM1 pour accéder à la table TEST1.EMP,
alors que le user TEST2 lancera l'exécution du programme PGM1 avec le package CTEST2.PGM1 pour accéder à la table TEST2.EMP.

N.B : affichage du contenu du DBRM et de la colonne STMT de la table SYSIBM.SYSPACKSTMT :

6 - Choix du package à l'exécution

Prenons le cas d'un source COBOL « moduA » qui est l'un des sous-modules appelés par call dynamique lors de l'exécution d'une « run-unit » démarrée par l'exécution du programme principal de nom **pgma**.



Lorsqu'on exécute le programme **pgma**, l'instruction COBOL CALL moduA lance le déroulement du code prévu dans le source moduA, et donc l'exécution des instructions SQL contenues dans le dbrm moduA**,** plus précisément l'exécution de la méthode d'accès déterminée lors du bind package de moduA. Mais il y a 2 versions du package moduA :

- CollX.moduA
- CollY.moduA

Pour que l'exécution du Load Module moduA fonctionne, il faut indiquer à DB2 dans quelle collection il faut prendre le package moduA.

Cette information est fournie à DB2 dans un **plan d'application** (le nom de ca plan sera fourni ou déterminé au moment du lancement du programme principal pgma).

Un plan d'application est créé ou mis à jour par la commande DB2 BIND PLAN.

7 - Le BIND PLAN

La commande DB2 « BIND PLAN » permet de construire un plan d'application.

Le plan est une liste dans laquelle on peut mettre :

- des noms de packages explicites : CollX.moduA

- des noms de collections génériques : CollZ.*

Un plan est identifié par un nom de plan sur 8 caractères.

La syntaxe de la commande BIND PLAN ressemble à ceci :

```
BIND PLAN (nom_du_plan)
     PKLIST (nom_coll1.nom_package1, nom_coll2.*, ...)
```

Lorsque DB2 a besoin d'exécuter les instructions SQL d'un module, il cherche ce module dans la liste « PKLIST » du plan :

- soit le module apparaît avec son nom explicite : CollX.moduA
- soit le nom du module n'apparaît pas explicitement, dans ce cas DB2 cherche parmi tous les modules de chacune des collections – dans l'ordre où sont listées les collections – jusqu'à ce qu'il trouve le bon module.

Remarques :

1) Quand faire le Bind plan

Si le plan référence globalement tous les packages d'une collection,
il n'est pas nécessaire de réexécuter la commande BIND PLAN lorsqu'on ajoute un nouveau package dans la collection par une commande bind package).

Tant que la pklist n'est pas modifiée, en cas de recompilation d'un module après une modification, il est totalement inutile de refaire le bind plan.

2) Privilèges sur le PLAN

Il peut être nécessaire, sur certains sites (suivant la valeur du owner fournie à la commande bind, …), de donner des privilèges sur le plan aux autres utilisateurs (commande SQL GRANT) de manière à permettre à ces autres utilisateurs d'utiliser le plan ou de faire des modifications sur cet objet : BIND PLAN pour modifier une PKLIST,

3) Bind de DBRMs directement dans un PLAN

Il est possible de relier directement un DBRM à un plan par une commande de la forme :

```
BIND PLAN (nom_du_plan)
    MEMBER (nom_de_dbrm)
    LIBRARY ('nom_de_biblio')
```

La commande Bind Plan peut contenir une liste de noms de packages et/ou de noms de collections et/ou la référence directe à des DBRM :

```
BIND PLAN (nom_du_plan)
     PKLIST (nom_coll1.nom_du_package1, nom_coll2.*, ...)
     MEMBER (nom_de_dbrm) LIB ('nom_de_biblio')
```

4) PKLIST

Il est possible dans la PKLIST du plan de faire apparaître le même module plusieurs fois dans des collections différentes : PKLIST(…,collX.dbrm1,… collY.dbrm1, …), en cas d'exécution du module dbrm1, si celui de la collection collX ne convient pas (version et/ou contoken ne correspondent pas à ceux demandés), la recherche se poursuit daans la pklist.

```
//*          BIND DU DBRM
//*
//BIND     EXEC PGM=IKJEFT01,COND=(4,LT,PC)
//STEPLIB  DD DISP=SHR,DSN=DSN510.DB2.DSNLOAD
//DBRMLIB  DD DISP=SHR,DSN=&LIBDBRM
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD  *
 DSN SYSTEM(DB2U)
  BIND PACKAGE(COL1) MEMBER(F01DBB0) -
       QUAL(EBXX001) -
       ACTION(REP)
 END
/*
```

//DBRMLIB DD …: définition de la librairie qui contient le DBRM à traiter, ont le nom de membre est indiqué par le paramètre MEMBER

Le DBRM est « bindé » dans la collection ‘COL1’.

Il peut être nécessaire de lancer une commande GRANT pour donner des privilèges sur le package à d’autres utilisateurs, on peut le faire dans le même step en insérant une instruction GRANT entre BIND et END, exemple :

```
  GRANT ALL ON PACKAGE COL1.F01DBB0 TO PUBLIC
```

Bind Plan

On donne ici le cas d’un plan générique qui peut être utilisé par plusieurs programmes. Il est inutile de faire le Bind Plan à chaque recompilation d’un programme.

Particularité : on a donné au plan le même nom que la collection qu’il référence, ce n’est pas du tout une obligation.

```
//BIND    EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=0M
//STEPLIB  DD  DISP=SHR,DSN=DSN510.DB2U.DSNEXIT
//         DD  DISP=SHR,DSN=DSN510.DB2.DSNLOAD
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//DBRMLIB  DD  DISP=SHR,DSN=TEST.FOR.DBRM
//SYSTSIN  DD  *
 DSN SYSTEM(DB2U)
  BIND PLAN(FORPL1) -
       PKLIST(COL1.*) -
       OWNER(EBXX001) -
       ACTION(REP)
 EN
 D
/*
```

Il peut être nécessaire de lancer une commande GRANT pour donner des privilèges sur le plan à d’autres utilisateurs, on peut le faire dans le même step en insérant une instruction GRANT entre BIND et END, exemple :

```
  GRANT BIND,EXECUTE ON PLAN COL1 TO PUBLIC
```

```
//************************************************
//*       EXECUTION D'UN PROGRAMME BATCH DB2 (RUN)    *
//************************************************
//RUN     EXEC PGM=IKJEFT01,DYNAMNBR=20
//STEPLIB  DD  DISP=SHR,DSN=DSN510.DB2.DSNLOAD
//         DD  DISP=SHR,DSN=TEST.FOR.LOADBA
//         DD  DISP=SHR,DSN=CEE.SCEERUN
//SYSTSPRT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSTSIN  DD  *
  DSN SYSTEM(DB2U)
  RUN  PROGRAM(F01DBB0) PLAN(FORPL1)
  END
/
```