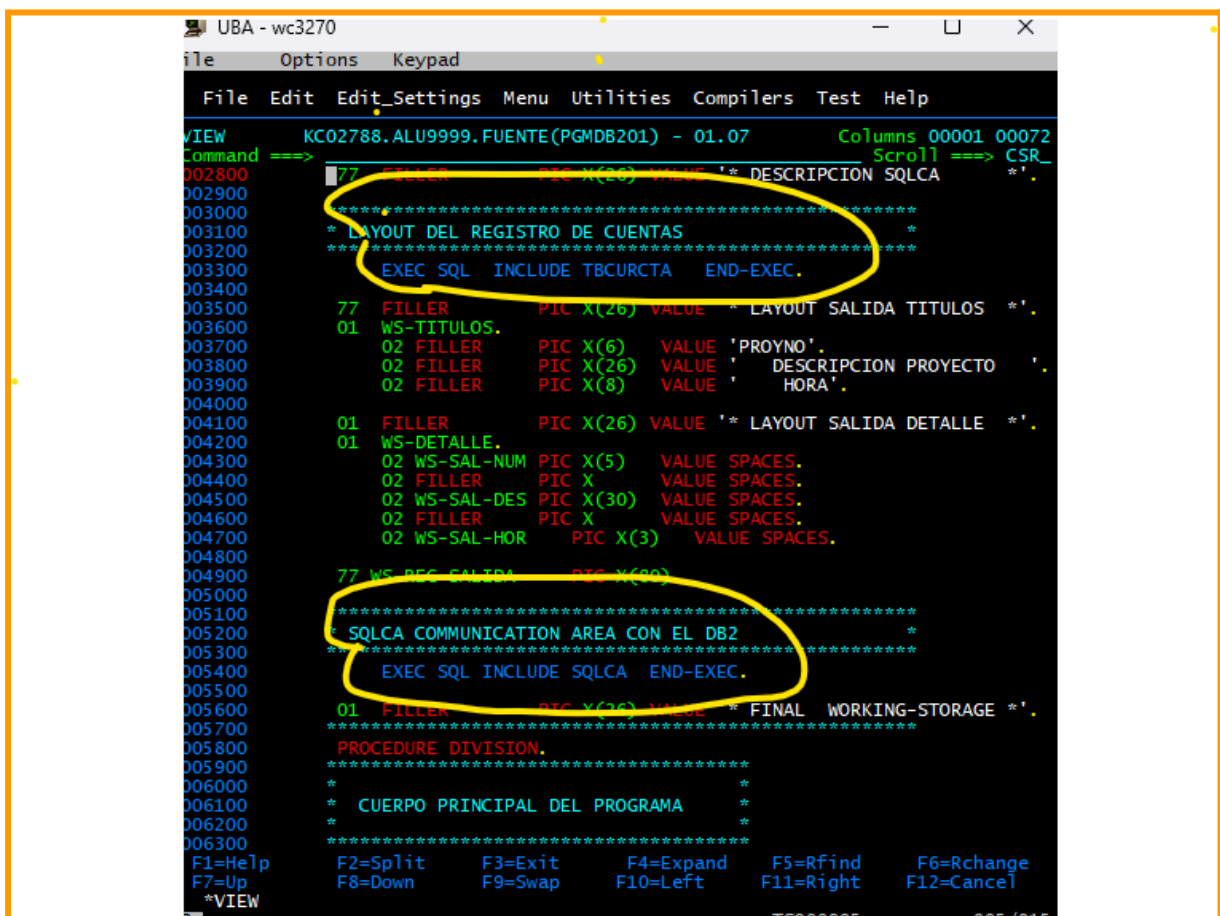


## OBJETIVO

- Estaremos recorriendo paso a paso el proceso de precompilación de un código COBOL con SQL embebido en su código.
- O sea que, con **Sequence Query Language** se acceden a los datos almacenados en bases de datos relacionales **DataBase 2**

## ESPECIFICACIONES

Preparando el código COBOL con SQL EMBEBIDO:



```

UBA - wc3270
File Edit Edit_Settings Menu Utilities Compilers Test Help
VIEW KC02788.ALU9999.FUENTE(PG MDB201) - 01.07 Columns 00001 00072
Command ==> Scroll ==> CSR_
002800 77 FILLER PIC X(26) VALUE '* DESCRIPCION SQLCA *'.
002900
003000 *****
003100 * LAYOUT DEL REGISTRO DE CUENTAS *
003200 *****
003300 EXEC SQL INCLUDE TBCURCTA END-EXEC.
003400
003500 77 FILLER PIC X(26) VALUE '* LAYOUT SALIDA TITULOS *'.
003600 01 WS-TITULOS.
003700 02 FILLER PIC X(6) VALUE 'PROYNO'.
003800 02 FILLER PIC X(26) VALUE 'DESCRIPCION PROYECTO'.
003900 02 FILLER PIC X(8) VALUE 'HORA'.
004000
004100 01 FILLER PIC X(26) VALUE '* LAYOUT SALIDA DETALLE *'.
004200 01 WS-DETALLE.
004300 02 WS-SAL-NUM PIC X(5) VALUE SPACES.
004400 02 FILLER PIC X VALUE SPACES.
004500 02 WS-SAL-DES PIC X(30) VALUE SPACES.
004600 02 FILLER PIC X VALUE SPACES.
004700 02 WS-SAL-HOR PIC X(3) VALUE SPACES.
004800
004900 77 WS-REC-SALIDA PIC X(80)
005000 *****
005100 * SQLCA COMMUNICATION AREA CON EL DB2 *
005200 *****
005300 EXEC SQL INCLUDE SQLCA END-EXEC.
005400
005500 01 FILLER PIC X(26) VALUE '* FINAL WORKING-STORAGE *'.
005600 *****
005700 PROCEDURE DIVISION.
005800 *****
005900 * CUERPO PRINCIPAL DEL PROGRAMA *
006000 *****
006100
006200
006300
F1=Help F2=Split F3=Exit F4=Expand F5=Rfind F6=Rchange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel
*VIEW
T5000005 005/015

```

## EXEC SQL INCLUDE SQLCA END-EXEC.

Se refiere al área de comunicación que TODO código COBOL con SQL embebido DEBE CONTENER.

## **EXEC SQL INCLUDE TBCURCTA END-EXEC.**

Se refiere a la tabla DB2 a la cual se va a acceder en nuestro caso: **KC02787.TBCURCTA**.

Pero para obtener el componente que se va a incluir; debemos ejecutar en forma online la herramienta: DCLGEN:

En el entorno Mainframe, **DCLGEN** (Declaration Generator) se refiere al utilitario provisto por IBM para simplificar el proceso de trabajo con tablas DB2 en programas. Resulta necesario generar las correspondientes declaraciones de estructura de tabla en SQL y de HOST VARIABLE STRUCTURES que resulten consistentes entre el programa y la estructura de la base de datos.

Para ello ingresaremos a través del ISPF Primary Option Menu a través de la opción:

- D2 DB2I**      Perform DB2 Interactive functions; luego
- 2 DCLGEN**      (Generate SQL and source language declarations)

```
UBA - wc3270
File Options Keypad
Menu Utilities Compilers Options Status Help
-----
ISPF Primary Option Menu
Option ==>
0 Settings      Terminal and user parameters      More: +
1 View          Display source data or listings
2 Edit          Create or change source data
3 Utilities      Perform utility functions
4 Foreground    Interactive language processing
5 Batch         Submit job for language processing
6 Command       Enter TSO or Workstation commands
7 Dialog Test   Perform dialog testing
8 LM Facility   Library administrator functions
9 IBM Products  IBM program development products
10 SCLM         SW Configuration Library Manager
11 Workplace    ISPF Object/Action Workplace

----- Other Install Products -----

SD SDSF        System Display and Search Facility
IP IPCS        Inter Problem Control Facility
IS ISMF        Inter Storage Management Facility
SM SMP/E       SMP/E and CBIPO Dialogs
HC HCD        HW Configuration Definition Dialog
R RACF        Resource Access Control Facility
S DFSORT       Data Facility Sort
OE OEDIT       OpenEdition MVS Edit files
OB OBROWSE     OpenEdition MVS Browse files
OS OSHELL      OpenEdition MVS ISPF Shell
BR READ        BookManager READ/MVS
BB BUILD       BookManager BUILD/MVS
BI READ INDEX  BookManager Index Utility
DA DXT ADMIN   Invoke DXT Administrative Dialogs
DE DXT END USER Invoke DXT End User Dialogs
M MVS/DITTO    MVS/DITTO Utility
IN INSPECT     INSPECT for C/3270 and PL/T
D2 DB2I        Perform DB2 Interactive functions
DM DB2ADM      DB2 Admin Tool
QM QMF         Query Management Facility
F1=Help      F2=Split    F3=Exit     F7=backward F8=Forward F9=Swap
F10=Actions  F11=Insert   F12=Insert
*DB2@PRI
TS000005      004/014
```

```

UBA - wc3270
file Options Keypad

DB2I PRIMARY OPTION MENU          SSID: DBDG

COMMAND ==>

Select one of the following DB2 functions and press ENTER.

1 SQLSTRT          (Process SQL statements)
2 DCLGEN           (Generate SQL and source language declarations)
3 PREPARE PREPARATION (Prepare a DB2 application program to run)
4 PRECOMPILE       (Invoke DB2 precompiler)
5 BIND/REBIND/FREE (BIND, REBIND, or FREE plans or packages)
6 RUN              (RUN an SQL program)
7 DB2 COMMANDS     (Issue DB2 commands)
8 UTILITIES        (Invoke DB2 utilities)
D DB2I DEFAULTS    (Set global parameters)
X EXIT             (Leave DB2I)

PRESS:                END to exit      HELP for more information

F1=HELP  F2=SPLIT  F3=END   F4=RETURN  F5=RFIND  F6=RCHANGE
F7=UP    F8=DOWN   F9=SWAP  F10=LEFT F11=RIGHT F12=RETRIEVE
*DSNEPRI

```

y completar los datos correspondientes a la estructura que se desea generar; en nuestro caso: **TBCURCTA**

Como se muestra en la siguiente imagen:

**1** - nombre de la tabla; en nuestro caso **TBCURCTA**

**2** - identificación de user id que generó dicha tabla, llamado OWNER de la misma; en nuestro caso: **KC02787**

**4** - nombre del dataset y miembro donde quedará alojada la salida de la estructura de tabla y correspondientes HOST VARIABLES

**6 - REPLACE:** recomendado porque lo hace re-ejecutable, o sea que, cada vez que se ejecuta reemplaza lo escrito en la ejecución anterior.

**9** - para ingresar un prefix en el nombre de las HOST VARIABLES: en nuestro caso: **WS-**

**11 - SUFFIX = YES** para que incluya el nombre del atributo de tabla en cada HOST VARIABLE

Resto de la configuración; según gusto y conciencia del programador.

```

UBA - wc3270
file Options Keypad

=====
DCLGEN                                SSID: DBDG

====>

Enter table name for which declarations are required:
1 SOURCE TABLE NAME ==> TBCURCTA

2 TABLE OWNER ..... ==> KC02787

3 AT LOCATION ..... ==>                                     (Optional)
Enter destination data set: (Can be sequential or partitioned)
4 DATA SET NAME ... ==> 'KC03XXX.CURSOS.DCLGEN(TBCURCTA)'
5 DATA SET PASSWORD ==>                                     (If password protected)
Enter options as desired:
6 ACTION ..... ==> REPLACE (ADD new or REPLACE old declaration)
7 COLUMN LABEL .... ==> NO (Enter YES for column label)
8 STRUCTURE NAME .. ==>                                     (Optional)
9 FIELD NAME PREFIX ==> WS- (Optional)
10 DELIMIT DBCS .... ==> YES (Enter YES to delimit DBCS identifiers)
11 COLUMN SUFFIX ... ==> YES (Enter YES to append column name)
12 INDICATOR VARS .. ==> NO (Enter YES for indicator variables)
13 ADDITIONAL OPTIONS==> NO (Enter YES to change additional options)

PRESS: ENTER to process   END to exit   HELP for more information

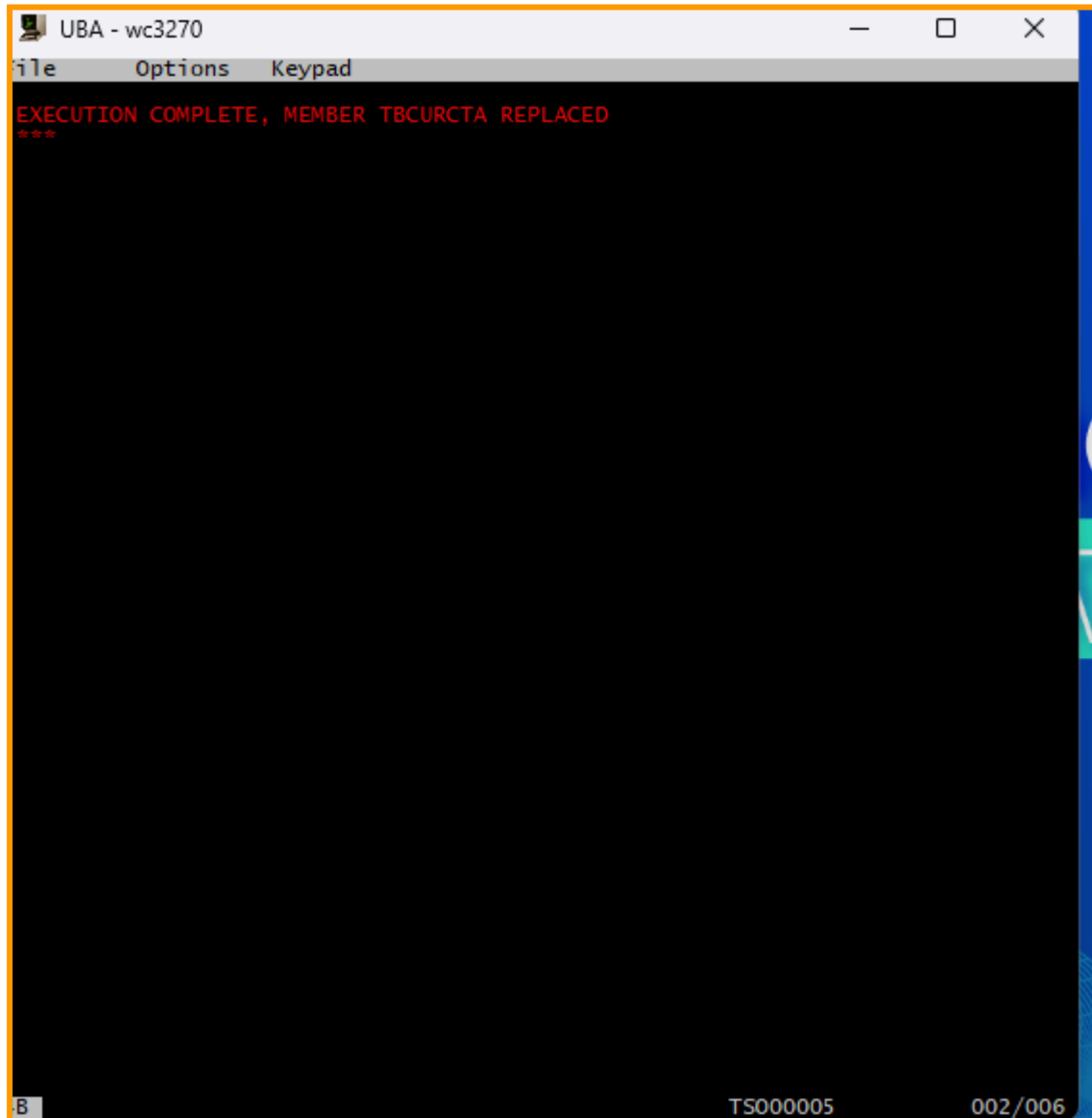
=====

DSNE294I  SYSTEM  RETCODE=000      USER OR DSN  RETCODE=0
F1=HELP   F7=UP      F8=DOWN    F9=SWAP    F10=LEFT   F11=RIGHT  F12=RETRIEVE
*DSNEDPO

```

La biblioteca PDS **KC03XXX.CURSOS.DCLGEN** donde XXX representan las últimas letras de cada user id; DEBE estar generada con la misma estructura que la biblioteca PDS: **KC03XXX.CURSOS.FUENTE**

Luego de ejecutado veremos esta imagen:



que significa que se generó el miembro correspondiente en biblioteca DCLGEN; en el ejemplo se generó el miembro TBCURCTA que será el que se incluya en el código fuente COBOL con SQL embebido.

Como se indica en la siguiente imagen:

```

UBA - wc3270
File Options Keypad
File Edit Edit_Settings Menu Utilities Compilers Test Help
VIEW KC02788.CURS05.DCLGEN(TBCURCTA) - 01.00 Columns 00001 00072
Command ==> Scroll ==> CSR_
***** Top of Data *****
000001 *****
000002 * DCLGEN TABLE(KC02787.TBCURCTA) *
000003 * LIBRARY(KC02788.CURS05.DCLGEN(TBCURCTA)) *
000004 * ACTION(REPLACE) *
000005 * LANGUAGE(COBOL) *
000006 * NAMES(WS-) *
000007 * QUOTE *
000008 * COLSUFFIX(YES) *
000009 * ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS *
000010 *****
000011 EXEC SQL DECLARE KC02787.TBCURCTA TABLE
000012 ( TIPCUEN CHAR(2) NOT NULL,
000013 NROCUEN DECIMAL(5, 0) NOT NULL,
000014 SUCUEN DECIMAL(2, 0) NOT NULL,
000015 NROCLI DECIMAL(3, 0) NOT NULL,
000016 SALDO DECIMAL(7, 2) NOT NULL,
000017 FECSAL DATE NOT NULL
000018 ) END-EXEC.
000019 *****
000020 * COBOL DECLARATION FOR TABLE KC02787.TBCURCTA *
000021 *****
000022 01 DCLTBCURCTA.
000023 * TIPCUEN
000024 10 WS-TIPCUEN PIC X(2).
000025 * NROCUEN
000026 10 WS-NROCUEN PIC S9(5)V USAGE COMP-3.
000027 * SUCUEN
000028 10 WS-SUCUEN PIC S9(2)V USAGE COMP-3.
000029 * NROCLI
000030 10 WS-NROCLI PIC S9(3)V USAGE COMP-3.
000031 * SALDO
000032 10 WS-SALDO PIC S9(5)V9(2) USAGE COMP-3.
000033 * FECSAL
000034 10 WS-FECSAL PIC X(10).
000035 *****
F1=Help F2=Split F3=Exit F4=Expand F5=Rfind F6=Rchange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel
*VIEW
.B TS000005 004/015

```



Por lo tanto cuando se escribe: EXEC SQL INCLUDE TBCURCTA END-EXEC.

Se estará incluyendo este miembro con la estructura de tabla TBCURCTA y las HOST VARIABLES asociadas. Las cuales se distinguen anteponiendo ‘:’ cuando se invocan en una query SQL.

Hasta acá estuvimos preparando el código COBOL con SQL EMBEBIDO.

### **Cómo nos preparamos para el PRECOMPILADOR?**

Definir el archivo PDS: **KC03XXX.CURSOS.DBRMLIB**  
a imagen del KC02788.ALU9999.DCLGEN

en esta nueva biblioteca o archivo PDS (PO) quedará la salida del miembro (código COBOL correspondiente) **COMPILADO OK**

---

```
UBA - wc3270
le  Options  Keypad

Data Set Information
Command ==>

Data Set Name . . . : KC02788.ALU9999.DBRMLIB

General Data
Management class . . : **None**
Storage class . . . : PRIM90
Volume serial . . . : KCTR50
Device type . . . : 3390
Data class . . . : **None**
Organization . . . : PO
Record format . . . : FB
Record length . . . : 80
Block size . . . : 27920
1st extent cylinders: 10
Secondary cylinders : 10
Data set name type : PDS
Data set encryption : NO

Current Allocation
Allocated cylinders : 10
Allocated extents . : 1

Current Utilization
Used cylinders . . : 1
Used extents . . . : 1

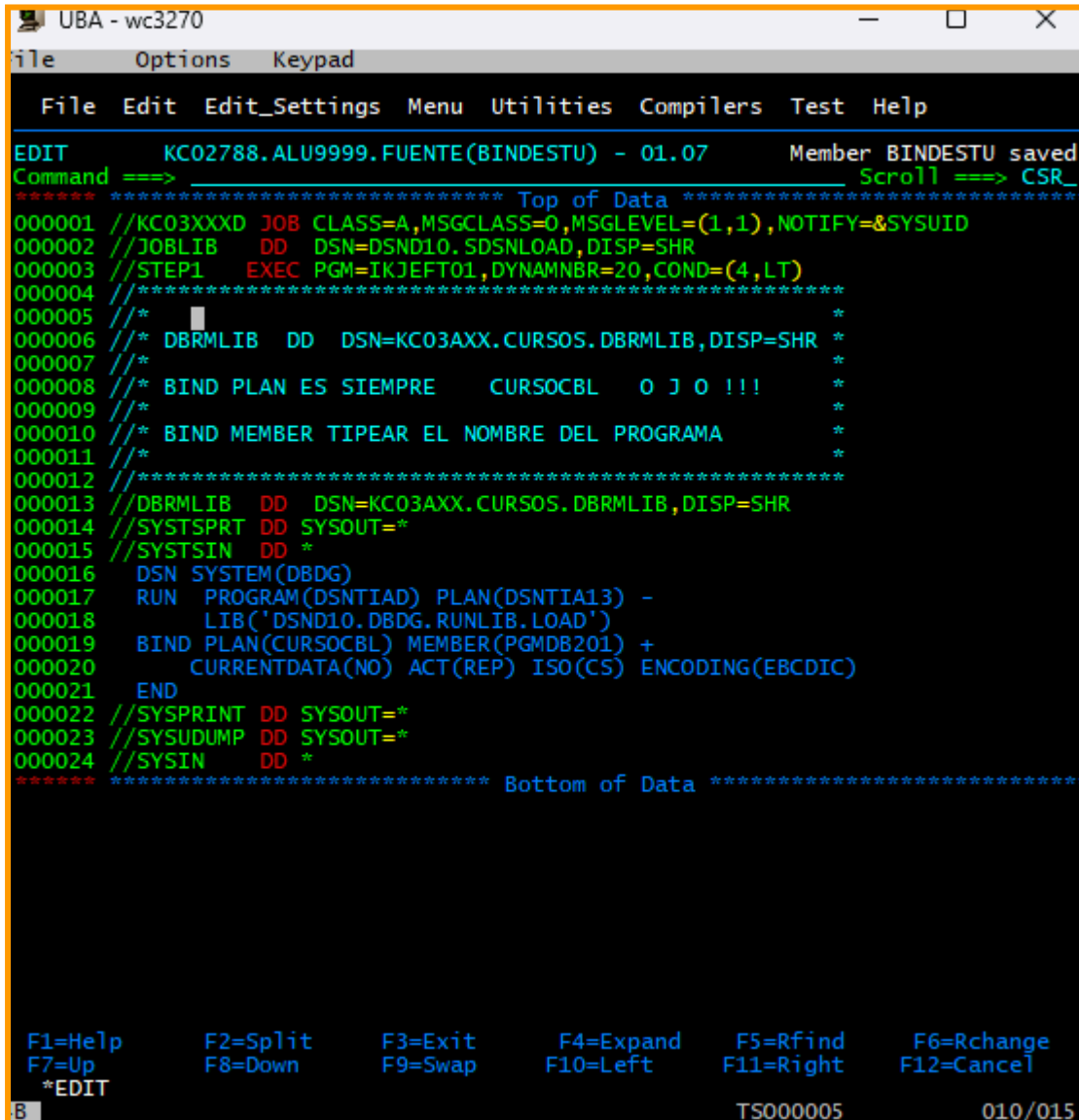
Dates
Creation date . . . : 2024/05/14
Referenced date . . : 2025/06/13
Expiration date . . : ***None***

F1=Help      F2=Split    F3=Exit      F7=Backward  F8=Forward   F9=Swap
F12=Cancel
*DSLIST

TS000005      002/015
```

Ya estamos en condiciones de ejecutar el compilador: **COMPDB2**

Luego de ejecutado OK; o sea que generó el código objeto correspondiente. Faltará el BIND:



```

UBA - wc3270
File Options Keypad
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT KC02788.ALU9999.FUENTE(BINDESTU) - 01.07 Member BINDESTU saved
Command ==> Scroll ==> CSR_
***** Top of Data *****
000001 //KC03XXD JOB CLASS=A,MSGCLASS=0,MSGLEVEL=(1,1),NOTIFY=&SYSUID
000002 //JOB LIB DD DSN=DSND10.SDSNLOAD,DISP=SHR
000003 //STEP1 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
000004 //*****
000005 //*
000006 //* DBRMLIB DD DSN=KC03AXX.CURSOS.DBRMLIB,DISP=SHR *
000007 //*
000008 //* BIND PLAN ES SIEMPRE CURSOCBL OJO !!! *
000009 //*
000010 //* BIND MEMBER TIPEAR EL NOMBRE DEL PROGRAMA *
000011 //*
000012 //*****
000013 //DBRMLIB DD DSN=KC03AXX.CURSOS.DBRMLIB,DISP=SHR
000014 //SYSTSPRT DD SYSOUT=*
000015 //SYSTSIN DD *
000016 DSN SYSTEM(DBG)
000017 RUN PROGRAM(DSNTIAD) PLAN(DSNTIA13) -
000018 LIB('DSND10.DBG.RUNLIB.LOAD')
000019 BIND PLAN(CURSOCBL) MEMBER(PGMDB201) +
000020 CURRENTDATA(NO) ACT(REP) ISO(CS) ENCODING(EBCDIC)
000021 END
000022 //SYSPRINT DD SYSOUT=*
000023 //SYSUDUMP DD SYSOUT=*
000024 //SYSIN DD *
***** Bottom of Data *****

F1=Help F2=Split F3=Exit F4=Expand F5=Rfind F6=Rchange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel
*EDIT
B TS000005 010/015
  
```

## ¿Qué es DBRM?

Ahora daremos una explicación más detallada de un concepto nuevo que surge de las sentencias SQL embebidas en código COBOL: **DataBase Request Module**

En IBM DB2, **DBRM es el Módulo de Solicitud de Base de Datos**.

Es un componente clave en el proceso de sentencias SQL embebidas dentro de programas de aplicación en código COBOL, particularmente en entornos de mainframe.

### Vayamos por pasos:

- Un DBRM es un archivo o módulo creado durante la precompilación de un programa escrito en código COBOL que contiene sentencias SQL embebidas.
- Contiene las sentencias SQL extraídas del código fuente, junto con metadatos sobre esas sentencias, como rutas de acceso y detalles de optimización.
- El DBRM se utiliza como entrada durante el proceso de enlace (BIND), donde se convierte en un paquete o plan de aplicación.

## Pasos clave que involucran DBRM:

1. Precompilación:
  - a. Cuando un programa con SQL embebido es precompilado, las sentencias SQL se separan del código fuente.
  - b. El precompilador genera el DBRM y un archivo de código fuente modificado (con SQL reemplazado por llamadas a Db2).
2. Vinculación o BIND:
  - a. El DBRM utiliza el utilitario BIND de vinculación de DB2, que crea un paquete o plan.
  - b. El proceso de vinculación (bind) determina las rutas de acceso óptimas para las sentencias SQL.
3. Ejecución:
  - a. En tiempo de ejecución, la aplicación utiliza el paquete o plan vinculado para ejecutar las sentencias SQL de manera eficiente.

### **Beneficios del DBRM:**

- Optimización: Ayuda a DB2 a optimizar la ejecución de SQL al analizar los caminos de acceso durante el proceso de vinculación.
- Separación de Conocimientos: Mantiene la lógica SQL separada de la lógica de la aplicación.
- Reutilización: Los paquetes creados a partir de DBRMs se pueden reutilizar en múltiples ejecuciones del programa.

Cuando se está trabajando con DB2 en un mainframe, entender los DBRMs es esencial para gestionar el rendimiento de SQL y asegurar una ejecución fluida de la aplicación.

---

## SECUENCIA DE PRECOMPILACIÓN

La secuencia de precompilación de un programa COBOL con SQL embebido implica varios pasos específicos para preparar el código antes de su compilación y ejecución.

A continuación, se detalla un esquema general del proceso:

### 1. Escritura del Código Fuente

- El programador escribe el código fuente en COBOL, incluyendo sentencias SQL embebidas dentro de bloques EXEC SQL ... END-EXEC.

**Ejemplo:**

**IDENTIFICATION DIVISION.**

PROGRAM-ID. EJEMPLO.

**DATA DIVISION.**

**WORKING-STORAGE SECTION.**

01 WS-NOMBRE PIC X(50).

**PROCEDURE DIVISION.**

**EXEC SQL**

SELECT NOMBRE

INTO :WS-NOMBRE

FROM EMPLEADOS

WHERE ID = 1

**END-EXEC.**

### 2. Precompilación

- **Objetivo:** El precompilador convierte las sentencias SQL embebidas en llamadas a funciones específicas de la base de datos DB2 (en nuestro caso).

- **Resultado:** Se genera un archivo intermedio en COBOL puro (sin SQL embebido) y un archivo adicional con información de enlace (como un archivo de bind o plan).

**Secuencia lógica:**

- **Entrada:** Programa código COBOL con SQL embebido
- **Salida:**
  - Archivo COBOL puro (Programa compilado OKI).
  - Archivo de enlace (Programa enlace o BIND con DB2).

### **3. Compilación del Código COBOL**

- El archivo COBOL generado por el precompilador se compila con el compilador COBOL estándar.

### **4. Enlace (Binding)**

- El archivo de enlace generado en la precompilación se utiliza para crear un plan o paquete en la base de datos.

### **5. Ejecución**

- Una vez compilado y enlazado, el programa ejecutable puede interactuar con la base de datos.
-

## Flujo de Precompilación COBOL con SQL Embebido

### Descripción de los Pasos

1. **Código Fuente COBOL + SQL:** El programador escribe el código COBOL con sentencias SQL embebidas (por ejemplo, `EXEC SQL ... END-EXEC`).
2. **Precompilador SQL:**
  - Extrae las sentencias SQL del código fuente.
  - Genera un módulo de acceso a la base de datos (como un DBRM en DB2).
  - Sustituye las sentencias SQL en el código COBOL por llamadas a rutinas específicas de la base de datos.
3. **Compilador COBOL:** Compila el código COBOL puro generado por el precompilador.
4. **Enlazador/Linker:** Combina el código objeto COBOL con las bibliotecas necesarias (incluyendo las de acceso a la base de datos).
5. **Programa Ejecutable:** El resultado final es un programa ejecutable que puede interactuar con la base de datos.



## EMBEDDED SQL

