

## OBJETIVO

- Utilizando la herramienta File Manager; construir datos de prueba en un archivo QSAM (Queued Sequential Access Method) es muy simple y en este documento haremos el paso a paso.

## ESPECIFICACIONES

### ¿Qué es un archivo QSAM?

Es un tipo de archivo secuencial al que se accede en orden, registro por registro. Es el más simple de los métodos de acceso en z/OS, ideal para procesamiento por lotes (batch) o cuando no necesitás acceso directo aleatorio.

### Estructura típica de un archivo QSAM

#### 1. DSORG=PS

Esto indica que el archivo es físico secuencial.

#### 2. RECFM (Record Format)

Define cómo están estructurados los registros:

- F** (Fixed) o **FB** (Fixed Blocked): longitud fija, todos los registros iguales.
- V** (Variable) o **VB** (Variable Blocked): cada registro puede tener una longitud distinta.

#### 3. LRECL (Logical Record Length)

Longitud del registro lógico. Por ejemplo, si es **80**, cada registro ocupa 80 caracteres si es fijo.

#### 4. BLKSIZE (Block Size)

Tamaño del bloque físico que contiene uno o más registros. Puede ser múltiplo de LRECL.

## 5. Registros

Cada registro es simplemente una línea de texto estructurada según cada campo del mismo.

.Por ejemplo: estructura de registro de archivo de NOVEDADES DE CLIENTES

KC02788.ALU9999.COPYLIB(TBVCLIEN)

En algunos casos se pide incluir un registro de cabecera o comienzo y otro centinela o de fin, aunque no es obligatorio. Algunos procesos batch los esperan.

### ¿Cómo se accede?

Desde COBOL, lo accedés secuencialmente, registro por registro, usando comandos como **READ**, **WRITE**, **OPEN**, **CLOSE**.

### ¿Qué es un archivo VSAM?

Un archivo **VSAM** (**V**irtual **S**torage **A**ccess **M**ethod) es un tipo avanzado de conjunto de datos utilizado en sistemas IBM z/OS para almacenar y acceder a registros de manera eficiente.

A diferencia de los archivos secuenciales tradicionales como QSAM; VSAM permite múltiples formas de organización y acceso a los datos.

### Tipos de archivos VSAM

#### 1. **KSDS (Key Sequenced Data Set)**

Los registros se almacenan y acceden mediante una *clave primaria*. Ideal para búsquedas rápidas por campos como número de cliente o ID.

#### 2. **ESDS (Entry Sequenced Data Set)**

Similar a un archivo secuencial, pero con ventajas de rendimiento y gestión. Los registros se almacenan en el orden en que se ingresan.

3. **RRDS (Relative Record Data Set)**

Acceso por número de registro relativo. Útil cuando sabés exactamente en qué posición está el dato.

4. **LDS (Linear Data Set)**

No tiene estructura de registros, se usa como almacenamiento de bytes. Es común en bases de datos como **DB2**.

### Características clave

- **Acceso directo y secuencial**  
Podés leer registros en orden o saltar directamente a uno específico.
- **Índices internos**  
VSAM mantiene estructuras de índice para búsquedas rápidas.
- **Mejor rendimiento**  
Usa buffers, control de espacio y técnicas de organización que lo hacen más eficiente que QSAM.
- **Gestión con IDCAMS** (programa utilitario)  
Se crean, borran y manipulan con utilidades como **DEFINE**, **DELETE**, **REPRO**.

Se puede pensar el VSAM como una evolución de los archivos secuenciales, con más inteligencia y flexibilidad para manejar grandes volúmenes de datos.

---

## QUÉ DIFERENCIAS HAY CON UN ARCHIVO VSAM?

### QSAM (Queued Sequential Access Method)

- Acceso secuencial: Los registros se leen o escriben en orden, uno tras otro.
- Simplicidad: Son archivos planos, ejemplo: reportes
- Posibles formatos de registro: Fijo (F, FB) o variable (V, VB).
- Limitaciones: No permite acceso directo ni indexado. Si querés el registro 100, tenés que leer los 99 anteriores.
- Uso habitual: Procesos batch simples, entrada/salida de datos lineales.

### VSAM (Virtual Storage Access Method)

- Acceso más flexible: Soporta acceso secuencia (ESDS), directo (RRDS) e indexado (KSDS).
- Tipos de datasets:
  - KSDS (Key Sequenced): acceso por clave.
  - ESDS (Entry Sequenced): similar a un archivo QSAM.
  - RRDS (Relative Record): acceso por número relativo de desplazamiento a partir del primer registro ingresado.
- Mejor rendimiento: Optimizado para grandes volúmenes y acceso eficiente.
- Gestión avanzada: Usa catálogos, buffers y estructuras internas más complejas.
- Uso habitual: Aplicaciones transaccionales (CICS), bases de datos (DB2).

### RESUMIENDO:

- **QSAM** es como una cinta de casete vs
- **VSAM** es como un disco con acceso aleatorio y organizado.

Para realizar algo simple y lineal, como es generar DATOS DE PRUEBA; el QSAM va bien.

Si se requiere velocidad, búsqueda por clave o actualización frecuente, VSAM es el recomendado.

## Crear un archivo QSAM en forma manual

### 1. Ingresar mediante menú ISPF a la opción Allocate

#### 1. Entrá a File Manager desde ISPF

Normalmente se accede con la opción **F** del menú de herramientas si está instalada.

#### 2. Seleccioná la opción para crear o editar un archivo de datos

Usualmente esta opción está en **1 – Data Set Utility**.

#### 3. Definí el nombre del dataset (ej: **KC03XXX.DATOS.QSAM**)

Elegí el tipo de organización **PS** (Physical Sequential).

#### 4. Especificá los atributos

Por ejemplo:

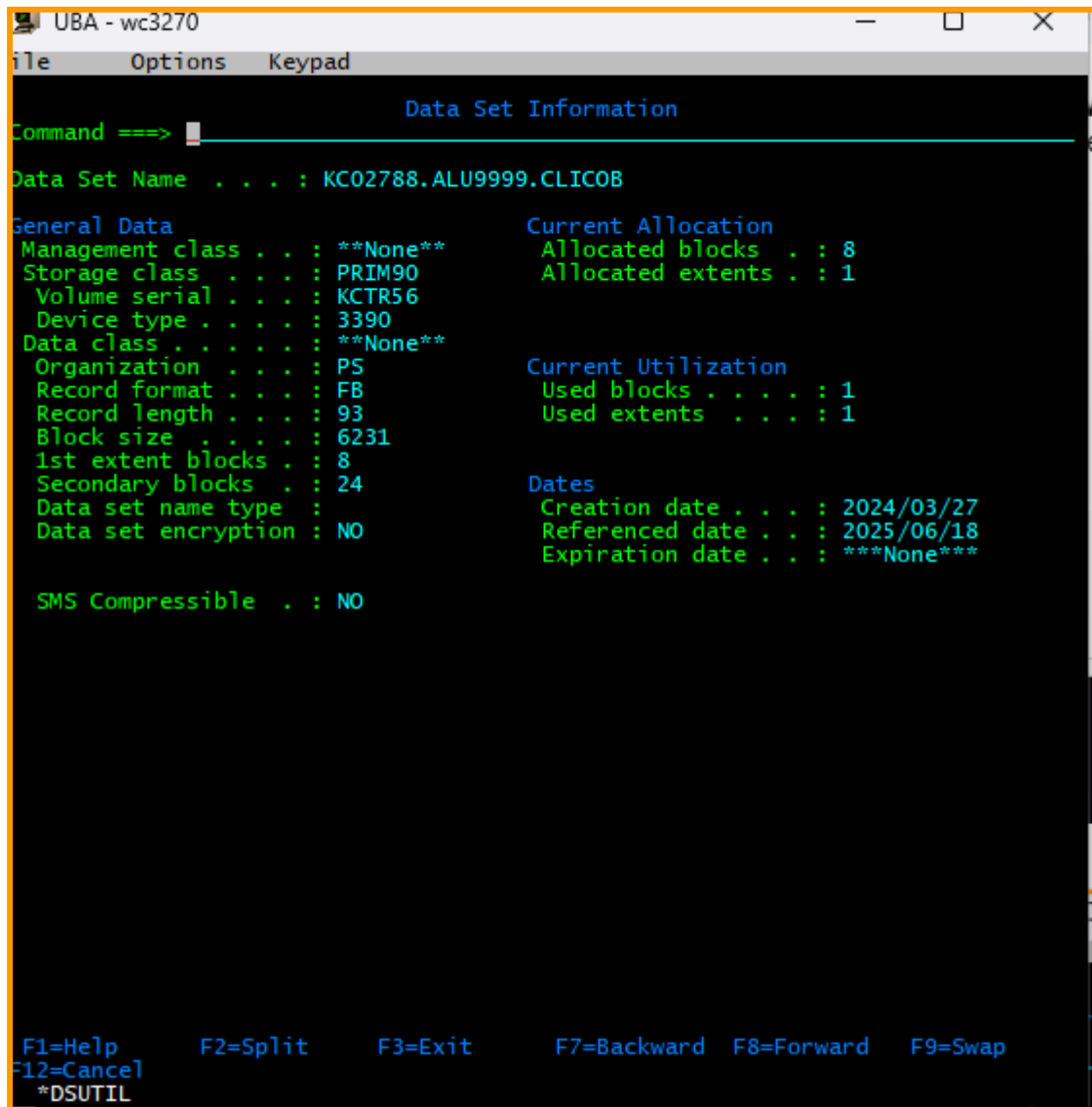
- LRECL: 244
- RECFM: FB
- BLKSIZE: 0 (dejalo en 0 para que el sistema lo calcule)

#### 5. Cargá los registros de prueba

Se puede hacer en forma manual; usando la **opción 2. EDIT del menú de FM**

- Como se muestra en la siguiente figura

En la figura de arriba se puede ver que se indica un **archivo QSAM existente** para poder copiar su estructura y repetir CASI TEXTUALMENTE al definir uno nuevo:



```
UBA - wc3270
File Options Keypad

Data Set Information

Command ==>

Data Set Name . . . : KC02788.ALU9999.CLIC08

General Data
Management class . . : **None**
Storage class . . . : PRIM90
Volume serial . . . : KCTR56
Device type . . . : 3390
Data class . . . : **None**
Organization . . . : PS
Record format . . . : FB
Record length . . . : 93
Block size . . . : 6231
1st extent blocks . : 8
Secondary blocks . : 24
Data set name type :
Data set encryption : NO

Current Allocation
Allocated blocks . : 8
Allocated extents . : 1

Current Utilization
Used blocks . . . : 1
Used extents . . . : 1

Dates
Creation date . . . : 2024/03/27
Referenced date . . : 2025/06/18
Expiration date . . : ***None***

SMS Compressible . : NO

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

Volvemos con tecla de función F3; y ahora ingresamos el **nombre del archivo QSAM NUEVO** y lo alocamos con el comando **A**:

```
UBA - wc3270
File Options Keypad
Menu RefList Utilities Help

Data Set Utility
Option ==> ____A____

  A Allocate new data set
  R Rename entire data set
  D Delete entire data set
blank Data set information
  C Catalog data set
  U Uncatalog data set
  S Short data set information
  V VSAM Utilities

ISPF Library:
Project . . . KC02788_
Group . . . ALU9999_
Type . . . FUENTE__

Enter "/" to select option
/ Confirm Data Set Delete

Other Partitioned, Sequential or VSAM Data Set:
Name . . . . . 'KC02788.NOVECLI'
Volume Serial . . . ____ (If not cataloged, required for option "C")

Data Set Password . . . (If password protected)

F1=Help      F2=Split      F3=Exit      F7=Backward  F8=Forward  F9=Swap
F10=Actions  F11=Insert   F12=Insert

*DSUTIL
.B TS000008 017/059
```

al presionar ENTER; en la siguiente pantalla, hacemos los cambios de largo de registro correspondientes; así como tener en cuenta de colocar 0 en el registro lógico o BLOCK SIZE:

```

UBA - WC3270
File Options Keypad
Menu RefList Utilities Help

Allocate New Data Set
Command ==>

Data Set Name . . . : KC02788.NOVECLI
Management class . . . (Blank for default management class)
Storage class . . . PRIM90 (Blank for default storage class)
Volume serial . . . KCTR50 (Blank for system default volume) **
Device type . . . (Generic unit or device address) **
Data class . . . (Blank for default data class)
Space units . . . BLOCK (BLKS, TRKS, CYLS, KB, MB, BYTES
or RECORDS)
Average record unit (M, K, or U)
Primary quantity . . 8 (In above units)
Secondary quantity . 24 (In above units)
Directory blocks . . 0 (Zero for sequential data set) *
Record format . . . FB
Record length . . . 244
Block size . . . 0
Data set name type (LIBRARY, PDS, LARGE, BASIC, *
Data set version . : EXTREQ, EXTPREF or blank)
Num of generations :
Extended Attributes (NO, OPT or blank)
Expiration date . . (YY/MM/DD, YYYY/MM/DD
YY.DDD, YYYY.DDD in Julian form
Enter "/" to select option
Dddd for retention period in days
or blank)

- Allocate Multiple Volumes

( * Specifying LIBRARY may override zero directory block)
( ** Only one of these fields may be specified)

F1=Help F2=Split F3=Exit F7=Backward F8=Forward F9=Swap
F10=Actions F11=Insert F12=Insert
*DSUTIL
.B TS000008 021/026

```

PRESIONAR 'ENTER':



```
UBA - wc3270
File Options Keypad
Menu RefList Utilities Help

Data Set Utility                                Data set allocated
option ==>

A Allocate new data set                        C Catalog data set
R Rename entire data set                      U Uncatalog data set
D Delete entire data set                      S Short data set information
Blank Data set information                    V VSAM Utilities

SPF Library:
Project . . . KC02788_                        Enter "/" to select option
Group . . . . ALU9999_                        / Confirm Data Set Delete
Type . . . . . FUENTE_

Whether Partitioned, Sequential or VSAM Data Set:
Name . . . . . 'KC02788.NOVECLI'
Volume Serial . . . . . (If not cataloged, required for option "C")
Data Set Password . . . . . (If password protected)

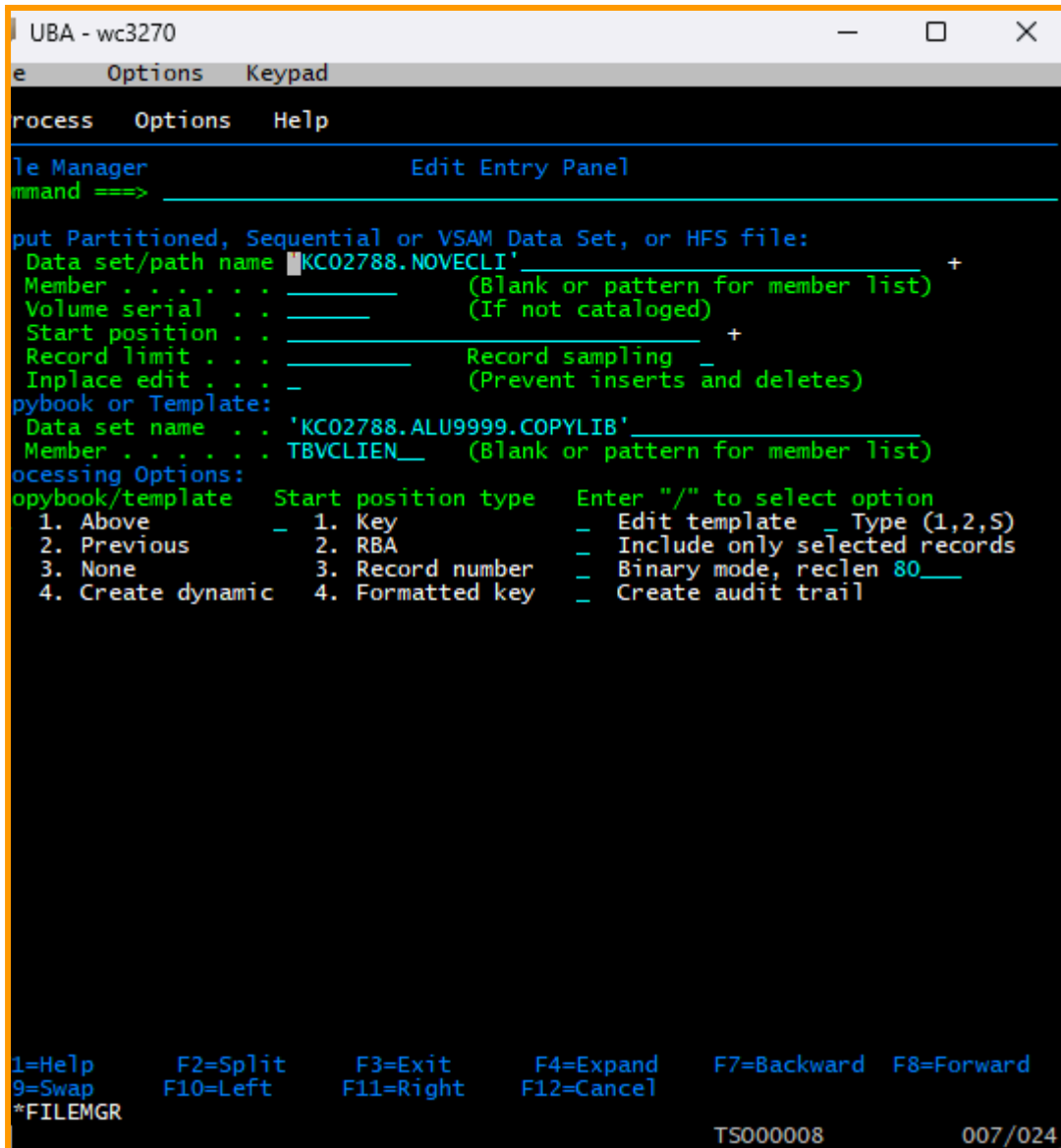
F1=Help    F2=Split    F3=Exit    F7=Backward  F8=Forward  F9=Swap
F10=Actions F11=Insert  F12=Insert
*DSUTIL

TS000008                                004/014
```

Ya hemos alocado el **nuevo archivo QSAM VACÍO**; sin registros de datos

Ahora estaremos cargando el contenido de los registros mediante File Manager OPCIÓN EDIT.

## Crear registros de datos en archivo QSAM mediante File Manager



```

UBA - wc3270
e Options Keypad
Process Options Help
File Manager Edit Entry Panel
Command ==>
Put Partitioned, Sequential or VSAM Data Set, or HFS file:
Data set/path name KC02788.NOVECLI' +
Member . . . . . (Blank or pattern for member list)
Volume serial . . (If not cataloged)
Start position . . +
Record limit . . . Record sampling _
Inplace edit . . . (Prevent inserts and deletes)
Copybook or Template:
Data set name . . 'KC02788.ALU9999.COPYLIB'
Member . . . . . TBVCLIEN_ (Blank or pattern for member list)
Processing Options:
Copybook/template Start position type Enter "/" to select option
1. Above - 1. Key - Edit template _ Type (1,2,5)
2. Previous - 2. RBA - Include only selected records
3. None - 3. Record number - Binary mode, reclen 80
4. Create dynamic 4. Formatted key - Create audit trail

F1=Help F2=Split F3=Exit F4=Expand F7=Backward F8=Forward
F9=Swap F10=Left F11=Right F12=Cancel
*FILEMGR TS000008 007/024
  
```

En la figura de arriba hemos escrito el nombre del archivos que generamos VACÍO con la estructura de registro de datos correspondiente para que el File Manager nos vaya guiando en la carga de cada CAMPO del registro.

Podremos generar uno a uno cada registro; en nuestro caso vamos a generar solamente 2 registros válidos.

Presionando ENTER vemos el archivo VACÍO:

```
UBA - wc3270
File Options Keypad
Process Options Help
Edit KC02788.NOVECLI Top of 0
Command ==> Record AT_TOP Scroll CSR_
WK-CLI-TIPO-NOVEDAD WK-CLI-TIPO-DOCUMENTO WK-CLI-NRO-DOCUMENTO WK-CLI-N
#2 #3 #4 #5
AN 1:2 AN 3:2 ZD 5:11 ZD 16:2
<> <---+---1-> <->
***** ***** Top of data *****
***** ***** End of data *****

F1=Help F2=Zoom F3=Exit F4=CRetriev F5=RFind F6=RChange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel
*FILEMGR TS000008 004/015
```

Agregando la opción I (insert) en cada línea de comando y presionando ENTER; podremos escribir el contenido de los campos:

```
UBA - wc3270
file Options Keypad
Process Options Help
Edit KC02788.NOVECLI Top of 1
Command ==> Record AT_TOP Scroll CSR_
WK-CLI-NACIONALIDAD WK-CLI-FECHA-DE-ALTA WK-CLI-FECHA-DE-BAJ
#12 #13 #14
AN 151:30 AN 181:10 AN 191:10
<---+---1---+---2---+---> <---+---> <---+--->
***** Top of data *****
000001 ARGENTINO 2025-06-21
***** End of data *****

F1=Help F2=Zoom F3=Exit F4=CRetrieval F5=RFind F6=RChange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel
*FILEMGR TS000008 004/015
```

Al presionar ENTER y PF3:

```
UBA - wc3270
File Manager      Edit Entry Panel      1 record(s) updated
Command ==> _____

Input Partitioned, Sequential or VSAM Data Set, or HFS file:
  Data set/path name 'KC02788.NOVECLI'_____ +
  Member . . . . . _____ (Blank or pattern for member list)
  Volume serial . . _____ (If not cataloged)
  Start position . . _____ +
  Record limit . . _____ Record sampling _____
  Inplace edit . . . - (Prevent inserts and deletes)
Copybook or Template:
  Data set name . . 'KC02788.ALU9999.COPYLIB'_____
  Member . . . . . TBVCLIEN__ (Blank or pattern for member list)
Processing Options:
Copybook/template  Start position type  Enter "/" to select option
1 1. Above         - 1. Key              - Edit template _ Type (1,2,S)
  2. Previous      - 2. RBA              - Include only selected records
  3. None          - 3. Record number   - Binary mode, reclen 80____
  4. Create dynamic 4. Formatted key  - Create audit trail

F1=Help      F2=Split      F3=Exit      F4=Expand      F7=Backward  F8=Forward
F9=Swap      F10=Left      F11=Right    F12=Cancel
*FILEMGR

3 TS000008 007/024
```

Al volver a ingresar para ver los datos:

```

UBA - wc3270
File Options Keypad
Process Options Help
Edit KC02788.NOVECLI Top of 2
Command ==> Record AT_TOP Scroll CSR_
WK-CLI-TIPO-NOVEDAD WK-CLI-TIPO-DOCUMENTO WK-CLI-NRO-DOCUMENTO WK-CLI-N
#2 #3 #4 #5
AN 1:2 AN 3:2 ZD 5:11 ZD 16:2
<> <---+---1-> <->
***** ***** Top of data *****
000001 AL PE 123 1
000002 AL PE 124 2
***** ***** End of data *****

F1=Help F2=Zoom F3=Exit F4=CRetriev F5=RFind F6=RChange
F7=Up F8=Down F9=Swap F10=Left F11=Right F12=Cancel
*FILEMGR TS000008 004/015

```

Estamos viendo los dos registros generados.

Entonces estamos en condiciones de utilizar este archivo QSAM para convertirlo en VSAM; que es como se pide en el programa de la clase 27 para ejercitar acceso a base de datos DB2 con SQL embebido en código COBOL; haciendo INSERT de estos registros que acabamos de generar.

### Cómo convertir un archivo QSAM en un archivo VSAM KSDS?

Utilizar como esqueleto de definición: KC02788.ALU9999.FUENTE(EJENOVKS)

Luego de hacer las adecuaciones de cada user id para las tarjetas de JCL; primeras líneas.

### Vayamos por pasos de JCL:

- 1) **PASO DELCLUS:** Mediante IDCAMS se eliminan los archivos que resultan de SALIDA en esta ejecución o corrida de JCL
- 2) **PASO DEFKSDS:** Mediante IDCAMS se DEFINE el Cluster VSAM KSDS: KC03XXX.NOVECLI.KSDS.VSAM
  - a) Con su área de datos: KC03XXX.NOVECLI.KSDS.VSAM.DATA
  - b) y su área de índice: KC03XXX.NOVECLI.KSDS.VSAM.INDX
- 3) **PASO SORT1:** Mediante utilitario SORT estamos ordenando los datos de INPUT del archivo QSAM: KC03XXX.NOVECLI según la clave definida en el VSAM: posición relativa 1 por 17 bytes de largo. El archivo de SALIDA de este SORT será: KC03XXX.NOVECLI.CL
- 4) **PASO DEFREPRO:** Mediante IDCAMS opción REPRO estaremos copiando el archivo NOVECLI ordenado: KC03XXX.NOVECLI.CL en el archivo VSAM **KC03XXX.NOVECLI.KSDS.VSAM**

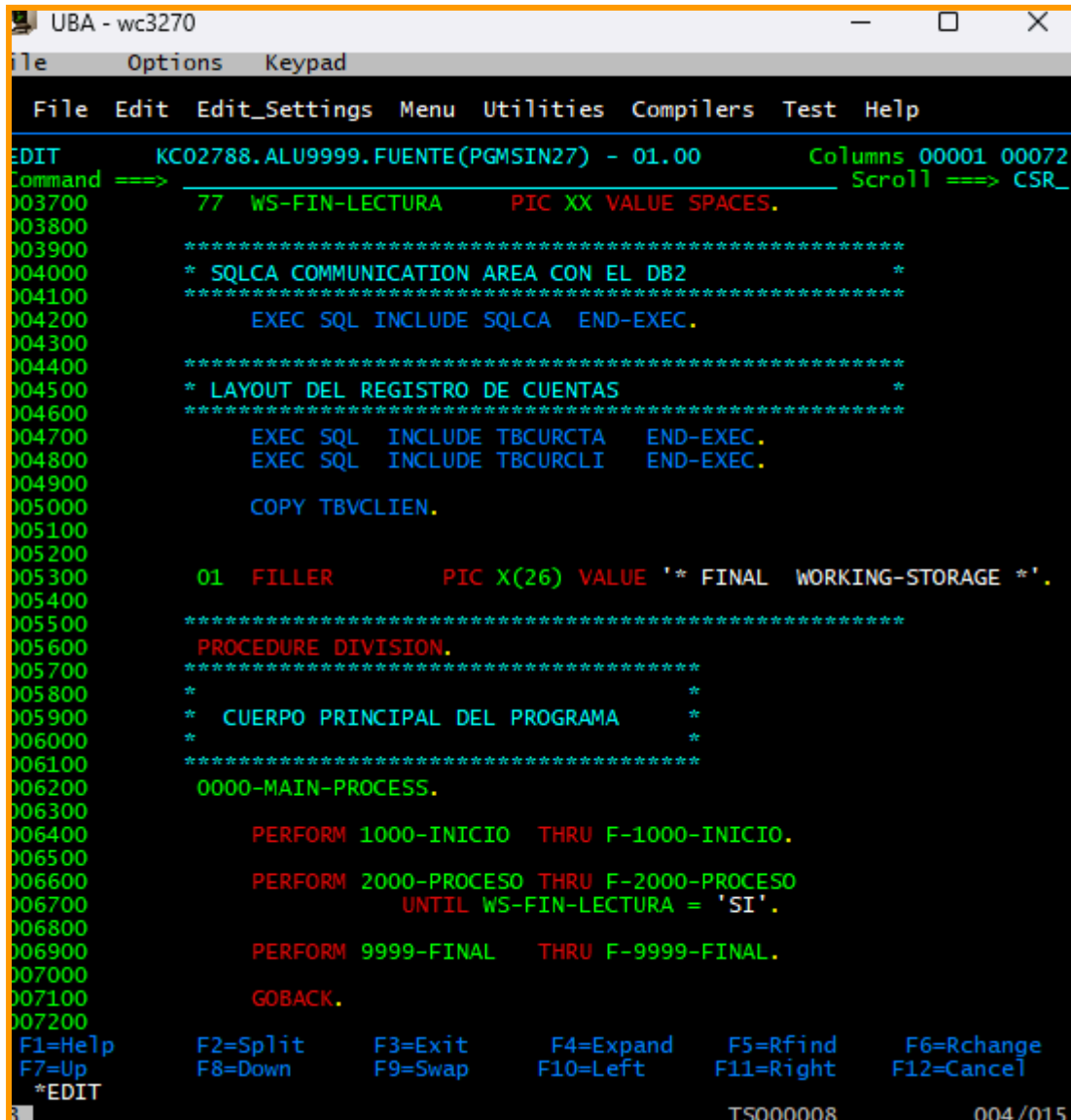
Ya hemos generado los datos de INPUT para el programa **PGMB2XXX** ; que resultó ser el **KC03XXX.NOVECLI.KSDS.VSAM**

Podrán ver el esqueleto del programa en: KC02788.ALU9999.FUENTE(PGMSIN27)

Podrán hacer las adecuaciones según nombres en cada DCLGEN: compilar y ejecutar BIND para relacionar las sentencias SQL con el programa COBOL.

## ¿Por qué necesitamos el DCLGEN????

En cada programa COBOL con SQL embebido encontraremos estas sentencias **INCLUDE**:



```

UBA - wc3270
File Options Keypad
File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT KC02788.ALU9999.FUENTE(PGMSIN27) - 01.00 Columns 00001 00072
Command ==> 77 WS-FIN-LECTURA PIC XX VALUE SPACES. Scroll ==> CSR_
003700
003800
003900
004000 *****
004100 * SQLCA COMMUNICATION AREA CON EL DB2 *
004200 *****
004300 EXEC SQL INCLUDE SQLCA END-EXEC.
004400
004500 *****
004600 * LAYOUT DEL REGISTRO DE CUENTAS *
004700 *****
004800 EXEC SQL INCLUDE TBCURCTA END-EXEC.
004900 EXEC SQL INCLUDE TBCURCLI END-EXEC.
005000
005100 COPY TBVCLIEN.
005200
005300 01 FILLER PIC X(26) VALUE '* FINAL WORKING-STORAGE *'.
005400
005500 *****
005600 PROCEDURE DIVISION.
005700 *****
005800 *
005900 * CUERPO PRINCIPAL DEL PROGRAMA *
006000 *
006100 *****
006200 0000-MAIN-PROCESS.
006300
006400 PERFORM 1000-INICIO THRU F-1000-INICIO.
006500
006600 PERFORM 2000-PROCESO THRU F-2000-PROCESO
006700 UNTIL WS-FIN-LECTURA = 'SI'.
006800
006900 PERFORM 9999-FINAL THRU F-9999-FINAL.
007000
007100 GOBACK.
007200
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
TS000008 004/015

```

La primera de ellas: **INCLUDE SQLCA** estará SIEMPRE, SIEMPRE, SIEMPRE en cada código COBOL con SQL embebido porque es el área de comunicación entre el programa y las sentencias de acceso a la base de datos DB2.



Las siguiente sentencias: **INCLUDE TBCURCTA** e **INCLUDE TBCURCLI** están referidas a la estructura de tabla y host variables (variables COBOL) que intervienen en cada código COBOL en particular.

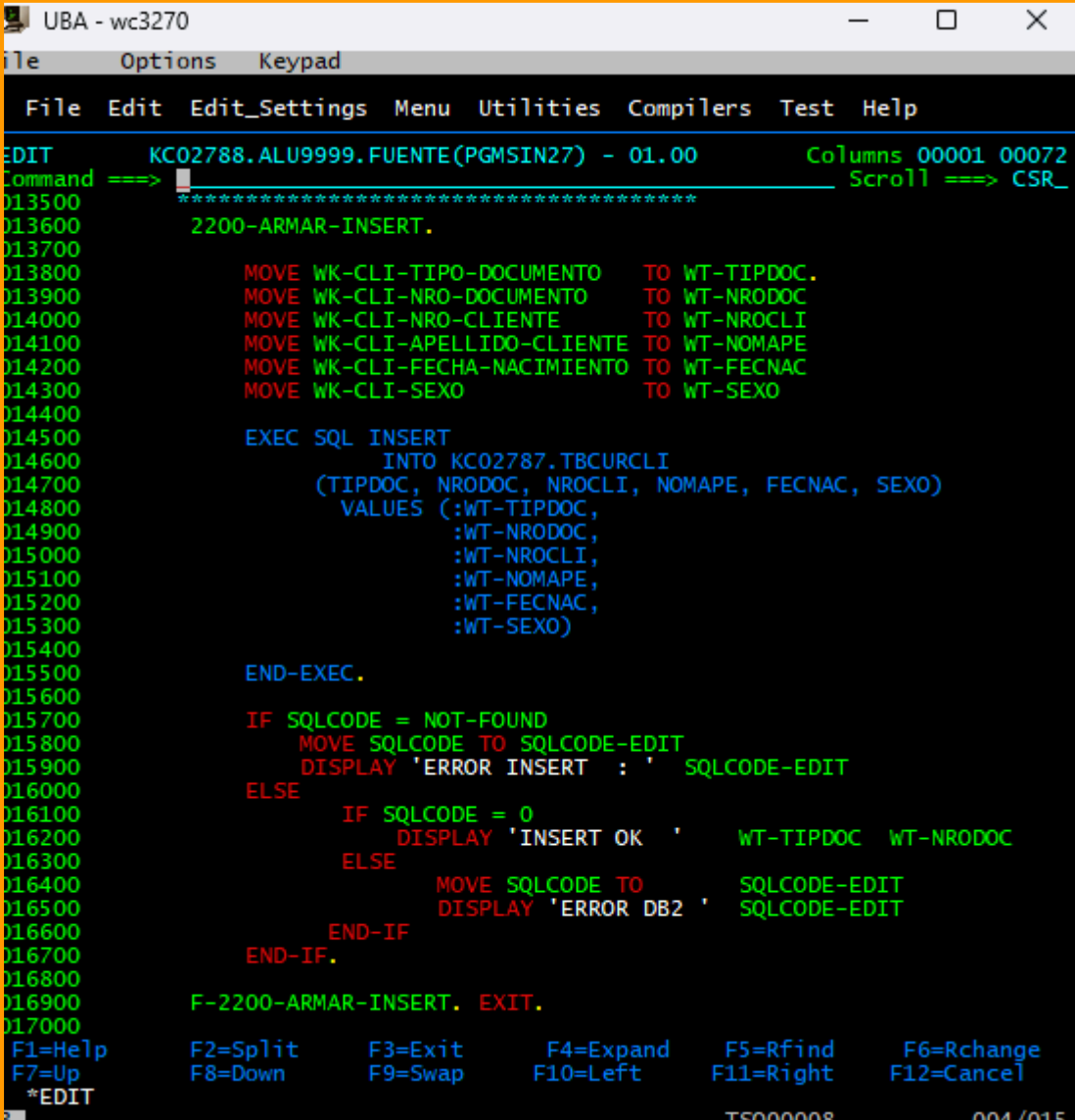
### **Ya hemos aprendido a ejecutar el DCLGEN:**

El generador de declaraciones de DB2: DCLGEN, genera las sentencias DECLARE necesarias para programas COBOL a partir de la estructura de CREATE de cada TABLA DB2 (en nuestro caso **KC02787.TBCURCLI** y **KC02787.TBCURCTA**).

**DCLGEN** genera también las estructuras de variable (HOST VARIABLES) de lenguaje COBOL correspondientes.

Cada salida de DCLGEN nos sirve a los programadores para incluirlas mediante SENTENCIA INCLUDE en el código COBOL que lo necesite.

Al tenerlas definidas en la WORKING STORAGE del programa COBOL con SQL embebido; podrán ser utilizadas dentro de las sentencias correspondientes en la PROCEDURE DIVISION; como se muestra en la siguiente figura:



```

EDIT      KC02788.ALU9999.FUENTE(PGMSIN27) - 01.00      Columns 00001 00072
Command ==>      Scroll ==> CSR_
013500      *****
013600      2200-ARMAR-INSERT.
013700
013800      MOVE WK-CLI-TIPO-DOCUMENTO      TO WT-TIPDOC.
013900      MOVE WK-CLI-NRO-DOCUMENTO      TO WT-NRODOC.
014000      MOVE WK-CLI-NRO-CLIENTE      TO WT-NROCLI.
014100      MOVE WK-CLI-APELLIDO-CLIENTE  TO WT-NOMAPE.
014200      MOVE WK-CLI-FECHA-NACIMIENTO  TO WT-FECNAC.
014300      MOVE WK-CLI-SEXO              TO WT-SEXO.
014400
014500      EXEC SQL INSERT
014600          INTO KC02787.TBCURCLI
014700          (TIPDOC, NRODOC, NROCLI, NOMAPE, FECNAC, SEXO)
014800          VALUES (:WT-TIPDOC,
014900                  :WT-NRODOC,
015000                  :WT-NROCLI,
015100                  :WT-NOMAPE,
015200                  :WT-FECNAC,
015300                  :WT-SEXO)
015400
015500      END-EXEC.
015600
015700      IF SQLCODE = NOT-FOUND
015800          MOVE SQLCODE TO SQLCODE-EDIT
015900          DISPLAY 'ERROR INSERT : ' SQLCODE-EDIT
016000      ELSE
016100          IF SQLCODE = 0
016200              DISPLAY 'INSERT OK '      WT-TIPDOC WT-NRODOC
016300          ELSE
016400              MOVE SQLCODE TO      SQLCODE-EDIT
016500              DISPLAY 'ERROR DB2 '  SQLCODE-EDIT
016600          END-IF
016700      END-IF.
016800
016900      F-2200-ARMAR-INSERT. EXIT.
017000
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
TS000008      004/015

```

Como se observa en la figura de arriba; se hace MOVE de cada campo del archivo VSAM de INPUT a cada atributo o columna de la tabla KC02787.TBCURCLI para poder realizar el INSERT solicitado en la práctica.

Nota importante: Observen que las fechas en tablas DB2 tienen formato DATE:

O SEA QUE: su estructura es: AAAA-MM-DD en 10 caracteres idéntico formato al VSAM de INPUT.

Si no fuera de esta manera habría que transformarlo a DATE antes del INSERT.

## REPASEMOS ALGUNOS DE LOS POSIBLES FORMATOS DE LOS ATRIBUTOS - EN DB2 -

En DB2, los **atributos de una tabla** (es decir, las columnas) pueden tener distintos formatos según el tipo de datos que almacenan. A continuación se expresan los formatos más comunes:

### Tipos de datos para atributos en DB2

- **CHAR**: Cadena de caracteres de longitud fija.
- **VARCHAR**: Cadena de caracteres de longitud variable.
- **INTEGER / SMALLINT**: Números enteros de distintos tamaños.
- **DECIMAL / NUMERIC**: Números con punto decimal fijo, ideales para cálculos financieros.

La ejecución de este código COBOL **PGMSIN27** está en: KC02788.ALU9999.FUENTE(EJESIN27)

AHORA A ELABORAR, COMPROMETERSE Y TRABAJAR, TRABAJAR Y TRABAJAR para poder COMPRENDER y FIJAR conocimientos.

Nada es magia, **TODO REQUIERE UN TIEMPO ESENCIAL DE PRÁCTICA**

En la próxima aprenderemos a trabajar con **CURSORES**, un nuevo amigo para acceder a varias filas de salida en la ejecución de una **query SQL..... embebido en código COBOL**.

Te acordás de la sentencia **FETCH** que vimos en la teoría?????

---