Programming Bases in Mainframe

Módulo Básico

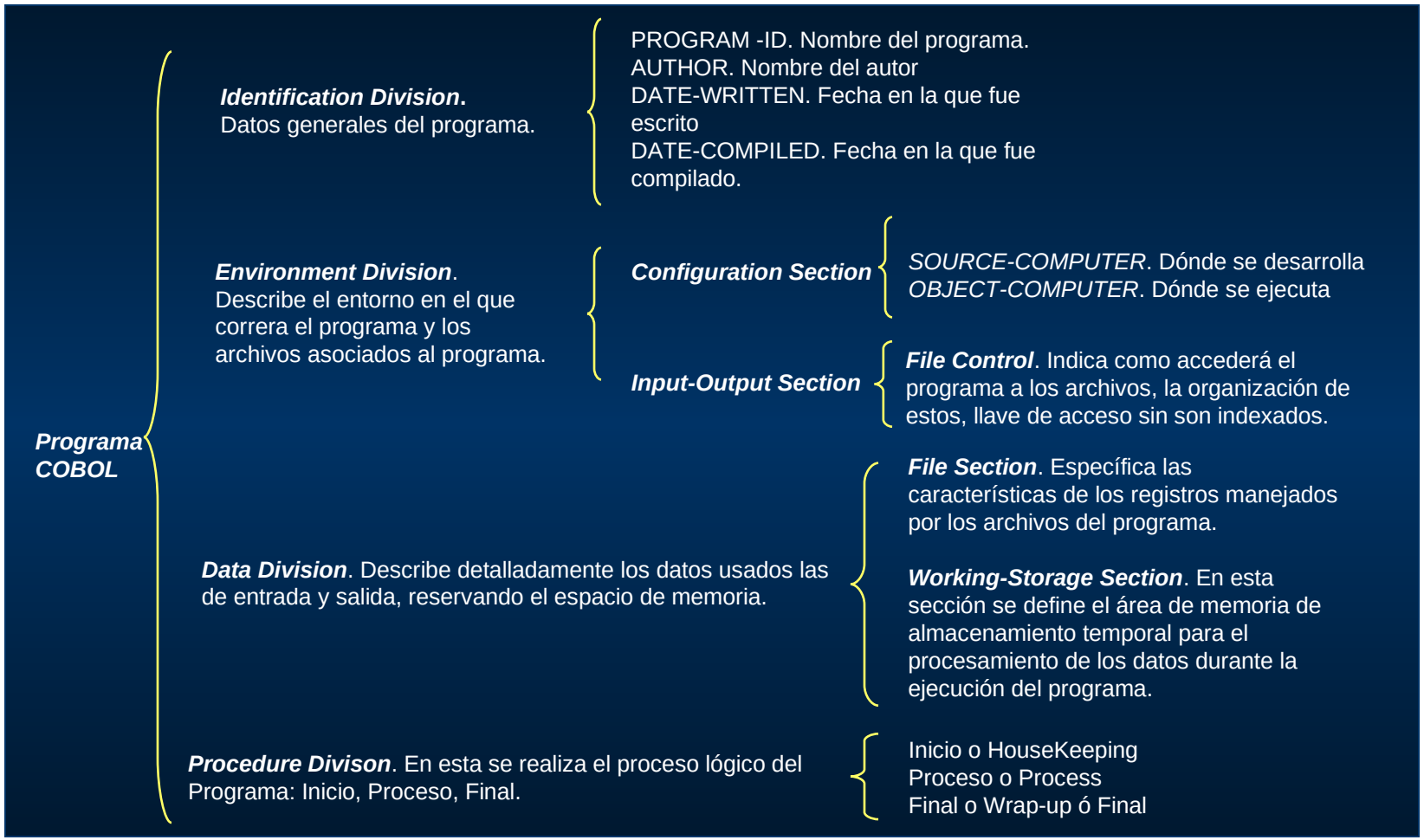
Módulo Básico

- **Tópico 310 - Coding a COBOL Program**
- Tópico 320 - Editing a Program
- Tópico 330 - File Processing
- Tópico 340 - Formatting Data
- Tópico 350 - COBOL Arithmetic and Program Logic
- Tópico 360 - Multimodular Programming and Copybooks
- Tópico 370 - Table Processing
- Tópico 380 - Galaxy Expense Processing System

Tópico 310 - Coding a COBOL Program

- OBJETIVOS.
  - Identificar las cuatro diferentes secciones y divisiones de COBOL.
  - Diferenciar entre un párrafo, una sentencia, una instrucción.
  - Codificar los comentarios de un programa COBOL.
  - Codificar las cuatro divisiones de un programa en COBOL.

Tópico 310 - Coding a COBOL Program



Tópico 310 - Coding a COBOL Program

- Working-Storage Section.
  - Record Layout.
    - Estructura de los cada uno de los campos que forman los registros de los archivos tratados en el programa.
  - Líneas y encabezados de reporte (lines and Headings).
    - Definición de constantes / variables que integran cada una de las líneas que corresponden a los títulos, cabeceras, líneas de detalle, totales, etc., de un reporte.
  - Flags/Switches.
    - Elementos de datos condicionales que indican verdadero o falso.
  - Contadores/Acumuladores.
    - Se refieren a una variable usada guardar el contenido del conteo o adición de elementos.

Tópico 310 - Coding a COBOL Program

- Working-Storage Section.
  - Constantes y literales.
    - Son variables que contienen un valor fijo, el cual no cambiará durante el proceso del programa. Por ejemplo: títulos, etiquetas de columnas, etc.
  - Variables de trabajo (Scratch Area).
    - Son variables que se encuentran involucradas en cálculos intermedios o en el proceso del programa. Por ejemplo: Una variable en la que en cada cambio de registro se vaya guardando el resultado de una operación.

Tópico 310 - Coding a COBOL Program

- Procedure Division.
  - Párrafo Mainline ó Principal.
    - Generalmente se le asigna el número de párrafo 1000.
    - Orden de ejecución de los procesos principales.
    - Manejo y control del programa principal, paso y recepción del control de otros programas.
    - Stop Run - finalización de programas.
  - Párrafo Housekeeping ó Inicio.
    - Generalmente se le asigna el número de párrafo 2000.
    - Incialización de variables de trabajo (acumuladores, contadores, etc).
    - En éste párrafo se realiza la apertura inicial de archivos

Tópico 310 - Coding a COBOL Program

- Procedure Division.
  - Párrafo Process ó Proceso**.**
    - Generalmente se le asigna el número de párrafo 3000.
    - Ejecución del procedimiento principal del programa
  - Párrafo Wraup-Up ó Final.
    - Generalmente se le asigna el número de párrafo 8000.
    - Cerrado de archivos usados durante la ejecución.

Tópico 310 - Coding a COBOL Program

- Párrafos principales

```
      ****************************************************************
       *                     PROCEDURE DIVISION                       *
       ****************************************************************
        PROCEDURE DIVISION.
        1000-MAINLINE.
            PERFORM 2000-HOUSEKEEPING
               THRU 2000-HOUSEKEEPING-EXIT.

            PERFORM 3000-PROCESS-TABLE
               THRU 3000-PROCESS-TABLE-EXIT.

            PERFORM 8000-WRAP-UP
               THRU 8000-WRAP-UP-EXIT.

            STOP RUN.
```

Tópico 310 - Coding a COBOL Program

```
*****************************************************************
*                        3100-LOAD-REG-EXP-DTLS                 *
*****************************************************************
 3100-LOAD-REG-EXP-DTLS.
           PERFORM 9910-READ-REG-EXP
              THRU 9910-READ-REG-EXP-EXIT.

           IF  NOT-END-OF-REG-EXP
               SET LS-TBL-IDX UP BY 1
               MOVE WS-REGE-REG-NUM  TO LS-TBL-REG-NUM(LS-TBL-IDX)
               MOVE WS-REGE-EXP-DT-J TO LS-TBL-EXP-DT-J(LS-TBL-IDX)
               MOVE WS-REGE-EXP-TIME TO LS-TBL-EXP-TIME(LS-TBL-IDX)
               MOVE WS-REGE-EXP-CODE TO LS-TBL-EXP-CODE(LS-TBL-IDX)
               MOVE WS-REGE-EXP-AMT  TO LS-TBL-EXP-AMT(LS-TBL-IDX)
           END-IF.

 3100-LOAD-REG-EXP-DTLS-EXIT.
     EXIT.
```

COMENTARIO

PRINCIPIO DE PÁRRAFO

SENTENCIA

INSTRUCCIONES

FIN DE DE PÁRRAFO

Tópico 310 - Coding a COBOL Program

- Párrafos en COBOL
  - Comentarios.
    - Se codifican a partir de columna 7 y comienzan con *
    - Máximo hasta la columna 72
  - Cabecera - inicio y finalización de un párrafo.
    - Se codifican a partir de la columna 8.
    - Nombrados de acuerdo a un orden lógico, máximo 35 caracteres
    - El párrafo de finalización marca la terminación del párrafo. El nombre debe tener el mismo nombre de inicio con terminación -EXIT.
    - El párrafo de terminación no es requerido.

Tópico 310 - Coding a COBOL Program

- Párrafos en COBOL
  - Sentencias
    - Colección de instrucciones que realizan un paso en el programa.
    - Se codifican a partir de la columna 12.
    - Una instrucción es un comando de COBOL (verbo) que realiza una función específica.
    - Las sentencias terminan con un punto, las instrucciones no.

Módulo Básico

- Tópico 310 - Coding a COBOL Program
- **Tópico 320 - Editing a Program**
- Tópico 330 - File Processing
- Tópico 340 - Formatting Data
- Tópico 350 - COBOL Arithmetic and Program Logic
- Tópico 360 - Multimodular Programming and Copybooks
- Tópico 370 - Table Processing
- Tópico 380 - Galaxy Expense Processing System

Tópico 320 - Editing a Program

- OBJETIVOS.
  - Conocimiento del software para la edición de programas.
  - Conocimiento de los comandos de edición.
  - Compilación y ejecución de un programa COBOL.
  - Edición y corrección de errores sintácticos y lógicos.

Tópico 320 - Editing a Program

- Codificación de programas
  - Comandos de edición
    - *Comandos en línea.*
      - Se teclean en la parte izquierda de la pantalla sobre la columna de números.

i - inserta líneas **i***n* - inserta el número de líneas especificado (*n*)

d - borra líneas **d***n* - inserta el número de líneas especificado (*n*)

c - copia la línea indicada **c***n* - copia el número de líneas indicado (*n*)

m- mueve la línea indicada **m***n* - mueve el número de líneas indicado (*n*)

r - repite la línea **r***n* - repite la linea el número de veces indicado

o - sobrescribe la línea **cols -** visualiza regla para posición de columnas.

Todos estos comandos con excepción de COLS, pueden ser utilizados con bloques de líneas, repitiendo el comando al inicio y al fin del párrafo.

Tópico 320 - Editing a Program

- Codificación de programas
  - Comandos de edición
    - *Comandos primarios.*
      - Realizan funciones de utilidad y se teclean en la línea de comando.
        - ✓ **CANCEL** - se sale del programa sin salvar los cambios
        - ✓ **SAVE** - Salva los cambios realizados al programa
        - ✓ **FIND** - Comando de búsqueda

          FIND '*carácter*' [ALL,FIRST,LAST,NEXT,PREV]
        - ✓ **CHANGE** - Sustitución de caracteres o cadenas de caracteres.

          CHANGE '*palabra_vieja*' '*palabra_nueva*' [ALL]

Tópico 320 - Editing a Program

- Codificación de programas
  - Comandos de edición
    - Comandos generales.
      - Teclas de función. Configurados de acuerdo con las teclas de función.
        - **PF1** - Ayuda de tutorial
        - **PF2** - División de pantallas
        - **PF3** - EXIT
        - **PF4** - Se sale de la actual operación y regresa al menú principal
        - **PF5** - Repite el último comando FIND
        - **PF6** - Ejecuta el comando CHANGE al ejecutar FIND
        - **PF7** - Desplaza la pantalla hacia arriba
        - **PF8** - Desplaza la pantalla hacia abajo
        - **PF10** - Desplaza la pantalla hacia la izquierda
        - **PF11** - Desplaza la pantalla hacia la derecha

Tópico 320 - Editing a Program

- Compilación de programas
  - Revisión de sintaxis.
    - Revisión de cada una de las sentencias escritas en el código fuentes
    - Depuración de errores
      - Errores de puntuación
      - Errores ortográficos
      - Errores de campos no definidos en Working Storage
      - Codificación de columnas inapropiadas
  - Generación de código objeto.
    - Se encuentra en lenguaje maquina.

Módulo Básico

- Tópico 310 - Coding a COBOL Program
- Tópico 320 - Editing a Program
- **Tópico 330 - File Processing**
- Tópico 340 - Formatting Data
- Tópico 350 - COBOL Arithmetic and Program Logic
- Tópico 360 - Multimodular Programming and Copybooks
- Tópico 370 - Table Processing
- Tópico 380 - Galaxy Expense Processing System

Tópico 330 - File Processing

- OBJETIVOS.
  - Generación de la sección de File Description y sentencias SELECT/ASSIGN para los archivos usados en el programa.
  - Codificación de la estructura de los registros de entrada y salida en la Working Storage.
  - Utilización de banderas en un programa COBOL.
  - Codificación de la Procedure Divsion en un programa.

Tópico 330 - File Processing

- Tipos de archivos
  - Archivos Secuenciales.
    - Organización
      - Deben ser accedidos en orden secuencial.
      - Para realización de búsquedas, son accedidos frecuentemente en un orden especifico, en base a un campo específico.
    - Acceso.
      - El acceso es secuencial es mas rápido que el acceso directo cuando se requiere de un procesamiento de todos los registros contenidos en el archivo, ya que el tiempo de procesamiento no involucra un acceso adicional al archivo del índice.
    - Dispositivos de almacenamiento.
      - Cualquier dispositivo (cintas, discos, etc).

Tópico 330 - File Processing

- Tipos de archivos *(Cont.)*
  - Archivos de Acceso Directo.
    - Deben ser accedidos a través de un índice.
    - Organización.
      - Son accedidos a través de un archivo especial llamado índice, el cual está formado por dos campos: un apuntador(dirección) y el campo llave.
    - Acceso.
      - Son de rápido acceso cuando se accede a través del campo llave.
      - No pueden ser accedidos desde cinta debido a la naturaleza de estos.
      - Su acceso puede ser secuencial ó indexado
    - Dispositivos de acceso.
      - Cualquier dispositivo de acceso, excepto cintas.

Tópico 330 - File Processing

- Environment Division.
  - Esta sección describe el entorno que requiere el programa, así como características de los archivos tales como : *nombre, localización y organización*.
  - Existen dos secciones:
    - ***Configuration Section***. Donde se especifican de forma informativa la computadora en se desarrolla y compila, así como la computadora donde se ejecuta:
      - SOURCE-COMPUTER.
      - OBJECT-COMPUTER.

Tópico 330 - File Processing

- Environment Division *(Cont.)*.
  - ***Input-Output Section.*** Describe las características de cada uno de los archivos que serán usados por el programa tanto de entrada como de salida. Esto es definido en la subsección de ***File Control***:
    - En archivos secuenciales:

      SELECT *nombre-interno-archivo*
      ASSIGN TO *nombre-externo*
      ORGANIZATION IS *estructura-archivo* [SEQUENTIAL]
      ACCESS MODE IS *acceso-archivo.* [SEQUENTIAL]

    - En archivos de acceso directo (indexados):

      SELECT *nombre-interno-archivo*
      ASSIGN TO *nombre-externo*
      ORGANIZATION IS *estructura-archivo* [SEQUENTIAL, INDEXED]
      ACCESS MODE IS *acceso-archivo* [RANDOM, SEQUENTIAL]
      RECORD KEY IS *llave-acceso*.

Tópico 330 - File Processing

- Data Division.
  - Reglas Nombres internos de archivos:
    - El nombre asignado a cada archivo debe ser único.
    - Pueden contener sólo letras, dígitos o guiones.
    - Deben contener al menos un carácter de A - Z.
    - No pueden contener guiones al inicio del nombre.
    - La longitud máxima del nombre es de 30 caracteres.
    - No puede ser una palabra reservada de COBOL

Tópico 330 - File Processing

- Data Division.
  - En esta división se define con mayor detalle las características de los archivos utilizados por los programas.
  - Esta división consta de dos secciones:
    - ***FILE SECTION.*** Contiene información de cada uno de los archivos de entrada y salida que requiere el programa de acuerdo con lo siguiente.

```
FD nombre-interno-archivo
    LABEL RECORDS ARE  tipo-dispositivo      [STANDARD, OMITTED]
    BLOCK CONTAINS n RECORDS                 [n, 0]
    RECORD CONTAINS n CHARACTERS             [n]
    DATA RECORD IS name-record.
01  name-record                             PIC X(##).
```

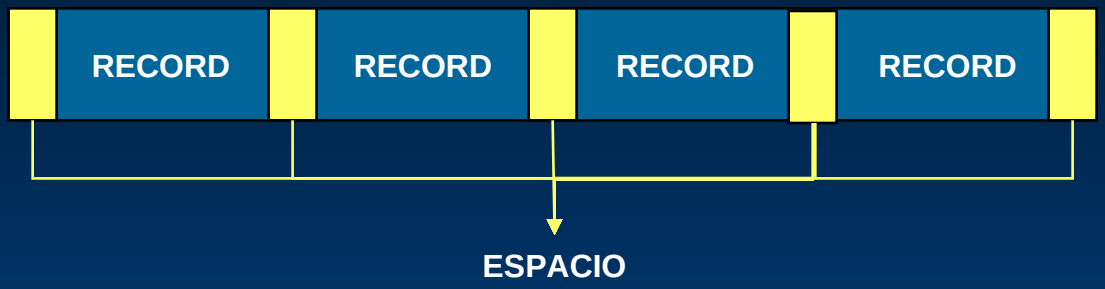
Tópico 330 - File Processing

- Data Division.
  - **FILE SECTION** (Cont.)
    - LABEL RECORDS ARE [STANDARD,OMITTED].
      - ✓ Es usada para indicar si se tratan de discos ó cintas (STANDARD), o bien, si el archivo será impreso (OMITTED).
    - BLOCK CONTAINS *n* RECORDS
      - ✓ Indica el número de registros que pueden ser leídos con un espacio entre cada registro (no bloqueados), o bien, en grupos con un espacio entre cada número determinado de registros (bloqueado).
      - ✓ Esta definición puede ser omitida, ya que sólo puede ser indicada para archivos bloqueados.
      - ✓ Se asigna 0 para que la que sea asignado por la maquina en el momento de la ejecución.
      - ✓ Para archivos que serán impresos se omite.

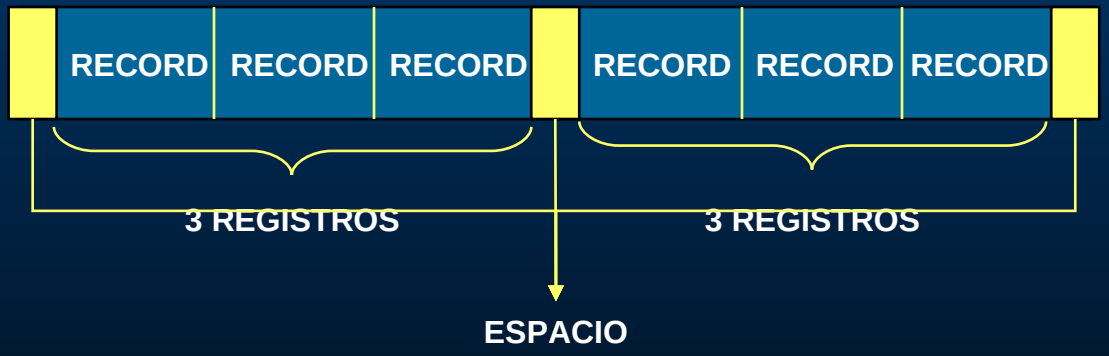
Tópico 330 - File Processing

- Data Division.

Registros no bloqueados



Registros bloqueados



Tópico 330 - File Processing

- Data Division.
  - **FILE SECTION (cont..)**
    - RECORD CONTAINS *n* CHARACTERS
      - ✓ Indica el número de caracteres por registro.
    - DATA RECORD IS *name-record*.
      - ✓ Indica el nombre del registro asociado al archivo, puede ser opcional.
      - ✓ La definición de este registro se encuentra en la Working-Storage Section, en la parte de Record Layout.
  - ***WORKING-STORAGE SECTION***. En esta sección se definen todos los datos que serán procesados por el programa y reserva espacio de memoria para dichos datos.

Tópico 330 - File Processing

- Definición de datos.
  - Definición de datos.
    - Los tipos de datos pueden ser:
      - Numéricos. Para números. Se identifica a través del caracter **9**.
      - Alfanuméricos. Para letras, números y símbolos especiales. Son identificados a través del caracter **X**.
      - Alfabéticos. Utilizado para letras. Son identificados a través del carácter **A**.
  - Los datos son definidos con la cláusula **PIC** seguida del tipo de dato y la longitud de este. Por ejemplo:

```
05  WS-NUMERO          PIC  9(25).
05  WS-NOMBRE                   PIC  X(25).
05  WS-NOMBRE         PIC  A(25).
```

Tópico 330 - File Processing

- Definición de datos.
  - La definición de registro de datos puede ser realizada a través de agrupación de datos, los cuales se encuentran definidos por niveles. Por ejemplo:

1er. Nivel
2do. Nivel
3er. Nivel

```
01 IN-EMP-REC.
    05 IN-EMP-NUM        PIC 9(5).
    05 IN-EMP-NAME
        10 IN-EMP-NAM    PIC X(6).
        10 IN-EMP-APP    PIC X(6).
    05 IN-EMP-CODE       PIC X(1).
```

Tamaño del registro = 18 bytes

Núm. Empleado	Nombre	C
12345	Molly Hayes	1

1	2	3	4	5	M	O	L	L	Y		H	A	Y	E	S		1

1 2 3 4 5 | 1 2 3 4 5 6 | 1 2 3 4 5 6 | 1

5 | 6 | 6 | 1

IN-EMP-REC

Tópico 330 - File Processing

- Definición de datos.
  - Décimales implícitos.
    - Este tipo de datos no reservan memoria para el punto décimal, con lo que no ocupa espacio.
    - El punto décimal implicíto es indicado con '**V**' en los campos numéricos. Por Ejemplo:

05 IN-EMP-CODE PIC 9(3)V99 ⟶ 51401

Tópico 330 - File Processing

- Secciones en Working-Storage Section.
  - Definición del registro (Record Layout).
    - Estructura de cada uno de los campos que forman los registros de los archivos tratados en el programa. Estos se encuentran mencionados a nivel de FD, en el nivel 01.

```
****************************************
*
*            DATA DIVISION
****************************************
*
 DATA DIVISION.
 FILE SECTION.
 FD  SEM-RPT-INPUT-FILE
     LABEL RECORDS ARE STANDARD
     BLOCK CONTAINS 0 RECORDS
     RECORD CONTAINS 37 CHARACTERS
     DATA RECORD IS INPUT-REC.
 01  INPUT-REC                  PIC X(37).
```

```
****************************************
*          WORKING-STORAGE SECTION
****************************************
 WORKING-STORAGE SECTION.
 01  WORKING-AREA.
****************************************
*               RECORD LAYOUTS
****************************************
      05 IN-SEMINAR-REC.
          10 IN-EMP-NUMBER          PIC 9(5).
          10 IN-EMP-NAME.
             15 IN-NAME-FIRST       PIC X(6).
             15 IN-NAME-LAST        PIC X(6).
          10 IN-EMP-CODE            PIC 9.
          10 IN-EMP-PHONE           PIC 9(7).
          10 IN-EMP-CONTACT         PIC X(12).
```

Tópico 330 - File Processing

- Secciones en Working-Storage Section.
  - Líneas y encabezados de reportes (Lines and Headings).
    - Definición de constantes /variables que integran cada una de las líneas que corresponden a los títulos, cabeceras, líneas de detalle, totales, etc., de un reporte.

```
*********************************************************
*                  WORKING-STORAGE SECTION
*********************************************************
 WORKING-STORAGE SECTION.
 01  WORKING-AREA.
*********************************************************
*                   REPORT LINES & HEADINGS
*********************************************************
    05 REPORT-LINES.
      10 RPT-HEADING1.
        15 FILLER     PIC X     VALUE SPACES.
        15 FILLER     PIC X(3)  VALUE 'RPT'.
        15 FILLER     PIC X(38) VALUE SPACES.
        15 FILLER     PIC X(19) VALUE 'TITULO'.
        15 FILLER     PIC X(71) VALUE SPACES.
```

Tópico 330 - File Processing

- Secciones en Working-Storage Section.
  - Flags/Switches.
    - Elementos de datos condicionales que indican verdadero o falso.
    - El nivel reservado para definir las condiciones de un switch es 88.
    - Cada cláusula debe ir acompañada con la palabra **VALUE**.

```
      ******************************************************
       *              WORKING-STORAGE SECTION               *
       ******************************************************
        WORKING-STORAGE SECTION.
        01  WORKING-AREA.
       ....
       ******************************************************
       *                  FLAGS AND SWITCHES                *
       ******************************************************
            05  SWITCHES.
             10  SW-END-OF-FILE          PIC X.
                 88  END-OF-FILE         VALUE 'Y'.
                 88  NOT-END-OF-FILE     VALUE 'N'.
```

Tópico 330 - File Processing

- Secciones en Working-Storage Section.
  - Flags/Switches (Cont.)
    - La condición de un switch es activada mediante la instrucción **SET**.

```
*****************************************************************
*                    9910-READ-REG-EXP                          *
*****************************************************************
9910-READ-REG-EXP.
    INITIALIZE INPUT-REC.
    READ INPUT-FILE INTO WS-REGE-REG-EXP
        AT END
            SET END-OF-FILE TO TRUE.
9910-READ-REG-EXP-EXIT.
    EXIT.
```

Tópico 330 - File Processing

- Secciones en Working-Storage Section.
  - Contadores / Acumuladores.
    - Se refieren a una variable usada guardar el contenido del conteo o adición de elementos.

```
      ************************************************************
      *                  WORKING-STORAGE SECTION                 *
      ************************************************************
       WORKING-STORAGE SECTION.
       01  WORKING-AREA.
       ....
      ************************************************************
      *                    COUNTERS AND ACCUMULATORS             *
      ************************************************************
           05  COUNTERS.
               10  CTR-CUENTA-LINEAS      PIC S9(2) COMP-3.
               10  CTR-CUENTA-PAGINAS     PIC S9(3) COMP-3.
           05  ACCUMULATORS.
               10  ACC-TOTAL-SALDO        PIC S9(2) COMP-3.
               10  ACC-TOTAL-CUENTA       PIC S9(3) COMP-3.
```

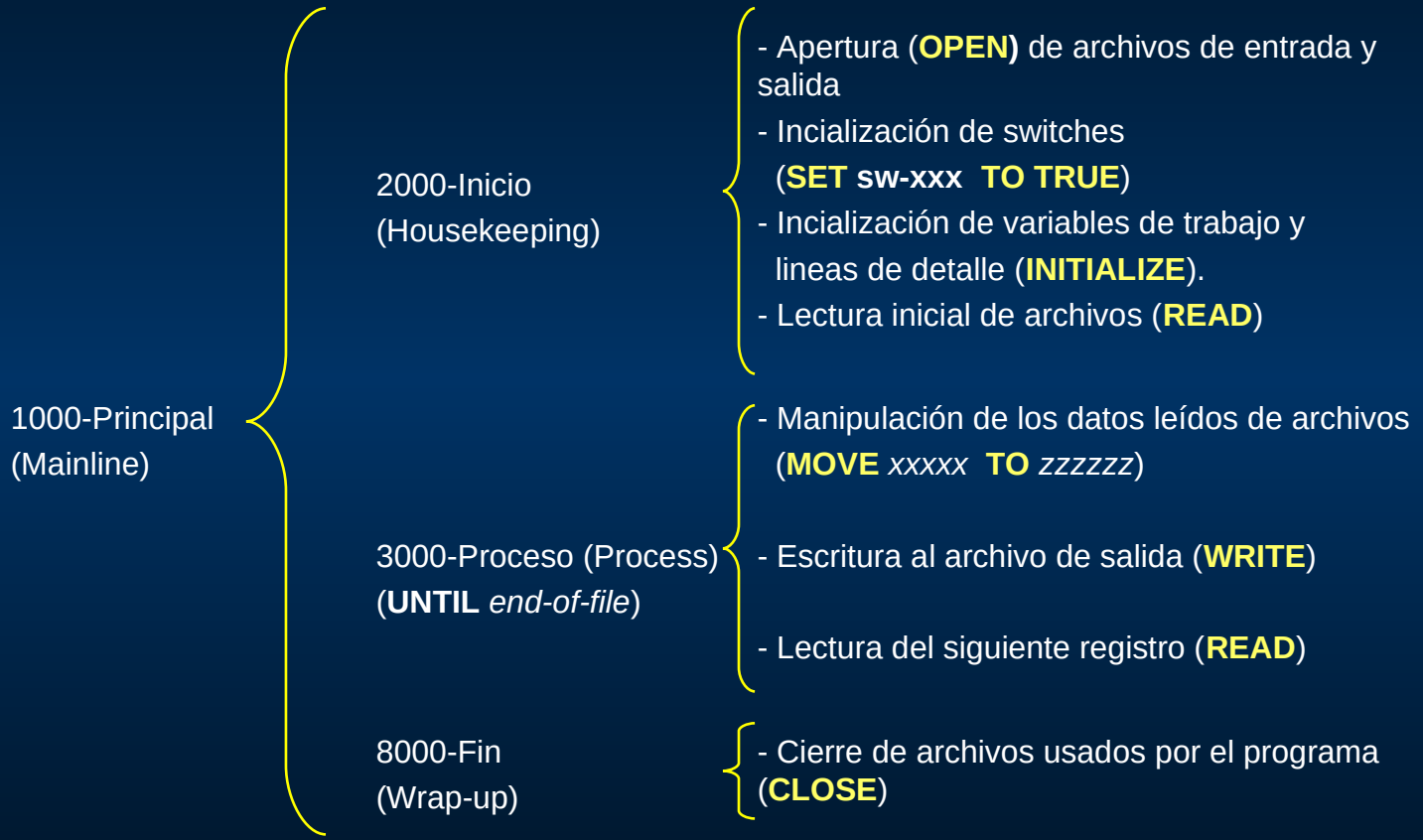
Tópico 330 - File Processing

- Secciones en Working-Storage Section.
  - Constantes.
    - Se refieren a una variable usada para guardar el contenido de un valor que nunca cambiará durante el proceso del programa.
    - Estos son indicados con la sentencia **VALUE**, después de haber especificado el PIC del contador o acumulador.
    - En caso de que la constante sea una literal ésta se pondrá entre comillas simples. Por ejemplo: 'PGM380'

```
      ************************************************************
      *                 WORKING-STORAGE SECTION                  *
      ************************************************************
       WORKING-STORAGE SECTION.
       01  WORKING-AREA.
      ************************************************************
      *                    LITERALS AND CONSTANTS                *
      ************************************************************
           05  CONSTANTS.
                10  CTE-ADD-ONE           PIC 9     VALUE 1.
                10  CTE-MAX-IDX           PIC 9(3)  VALUE 100.
```

Tópico 330 - File Processing

- Diagrama de Proceso para procesamiento de archivos y generación de listados



Tópico 330 - File Processing

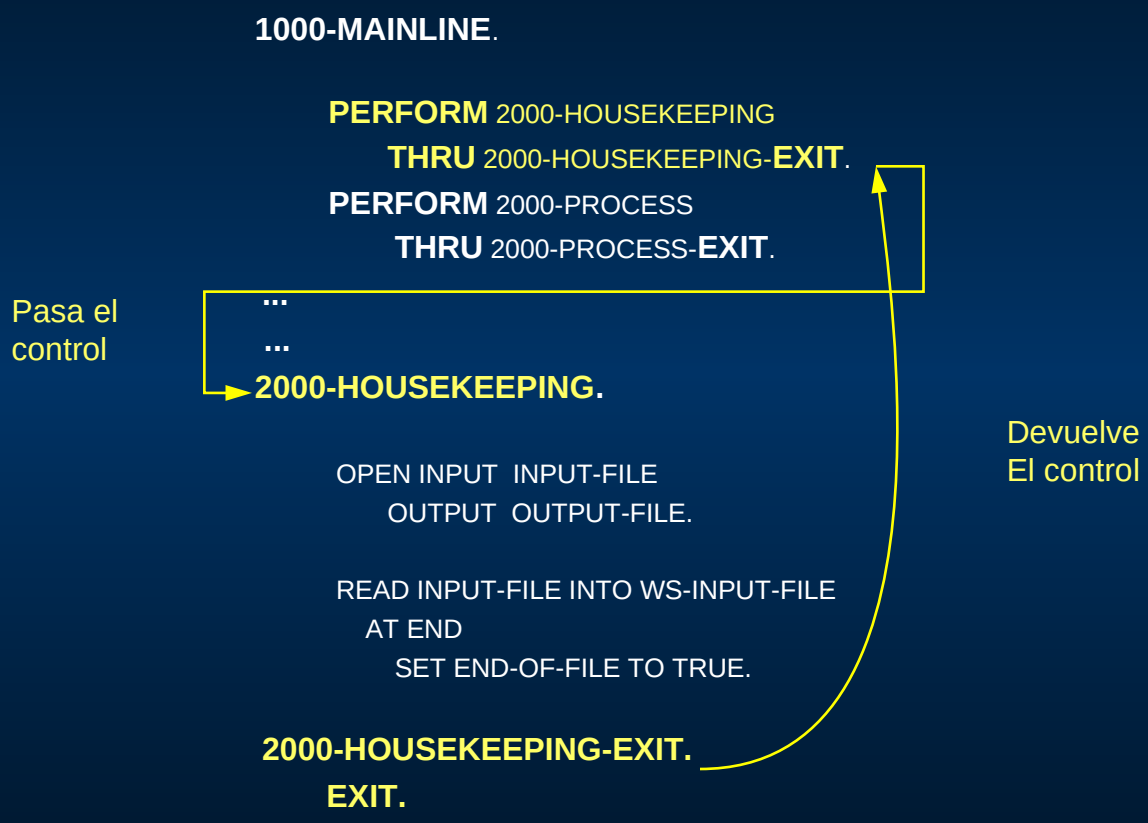
- Sentencia: **PERFORM**
  - Su función es pasar el control a un párrafo específico. Después de ejecutar las instrucciones de aquel párrafo, el control es devuelto al párrafo que realizó la llamada.
  - Esta sentencia es acompañada de la cláusula **THRU**, la cual permite identificar la última línea a ejecutar del párrafo.
  - Para indicar el final del párrafo se utiliza la cláusula **EXIT**.

PERFORM *nombre-parrafo*

THRU *nombre-párrafo*-**EXIT.**

Tópico 330 - File Processing

- Sentencia: **PERFORM** *(Cont.)*



Tópico 330 - File Processing

- Sentencia: **PERFORM**
  - La cláusula **UNTIL** es utilizada por la sentencia PERFORM para indicar que el párrafo debe ser ejecutado hasta cumplir una determinada condición.

PERFORM *nombre-parrafo*
THRU *nombre-parrafo*-**EXIT**
UNTIL *condicion*.

Por ejemplo:

PERFORM *2100-INIT-TABLE*
THRU *2100-INIT-TABLE*-***EXIT***
UNTIL *WS-CONTADOR EQUAL TO CTE-CINCO*.

Tópico 330 - File Processing

- Sentencia: **OPEN**
  - La sentencia **OPEN** permite abrir los archivos que requiere un programa además de :
    - Indicar al sistema cuáles son los archivos que serán accedidos por el programa
    - Identificar cuáles archivos serán usados para entrada de datos y cuáles serán archivos de salida de datos.
  - Generalmente es codificada al inicio del programa**.**
  - Su sintaxis es:

Archivos de lectura → **OPEN INPUT** *nombre-archivo*

Archivos de escritura → **OPEN OUTPUT** *nombre-archivo*

Adición de registros → **OPEN EXTEND** *nombre-archivo*

Tópico 330 - File Processing

- Sentencia: **OPEN**
  - En caso de tener varios archivos de lectura o escritura se codificaría así:

LECTURA	ESCRITURA
OPEN INPUT *File-name1* *File-name2* *File-name3* . . *File-namen*.	OPEN OUTPUT *File-name1* *F ile-name2* *File-name3* . . *File-namen*.

LECTURA-ESCRITURA	
OPEN INPUT	*File-name1*
	File-name2
	File-name3
OUTPUT	*File-nameX*
	.
	.
	File-namen.

Tópico 330 - File Processing

- Sentencia: **INITIALIZE**
  - Permite la inicialización de campos ya sea de grupo o elementales.
  - Los campos alfanuméricos son inicializados con espacios.
  - Los campos numéricos son inicializados con ceros.
  - Su sintaxis:

INITIALIZE *nombre-campo*

```
05 WS-CLIENTE.
     10  WS-CODIGO     PIC 9(5).
     10  WS-NOMBRE     PIC X(30).
```

```
INITIALIZE  WS-CLIENTE.
            ó
INITIALIZE WS-CODIGO
           WS-NOMBRE.
```

Tópico 330 - File Processing

- Sentencia: **MOVE**
  - La sentencia **MOVE** coloca una copia del dato contenido en un campo-1 a un campo-2.
  - Su sintaxis:

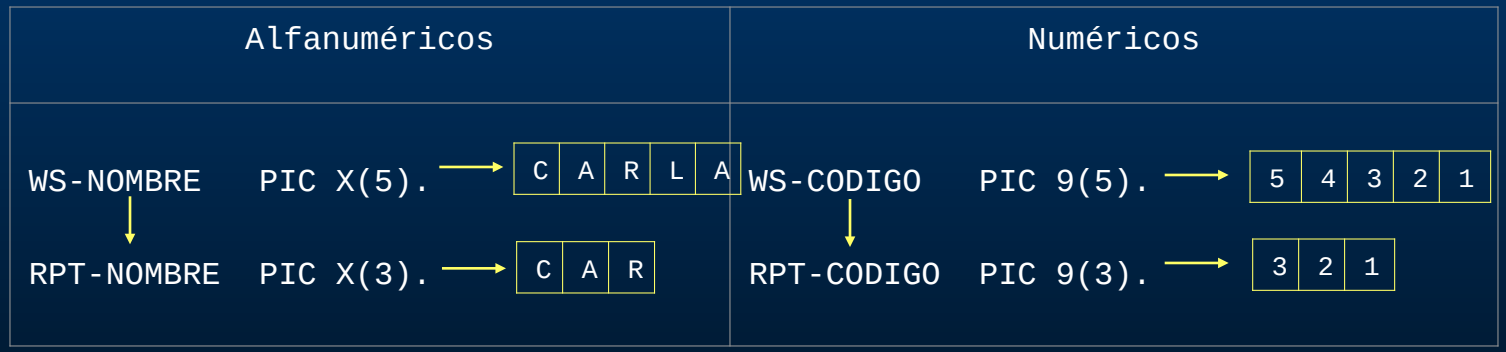
MOVE campo-1 **TO** campo-2

Por ejemplo:

Campos elementales	Campos a nivel grupo
MOVE campo-1 **TO** campo-2 campo-3 campo-4.	**MOVE** grupo-1 **TO** grupo-2.

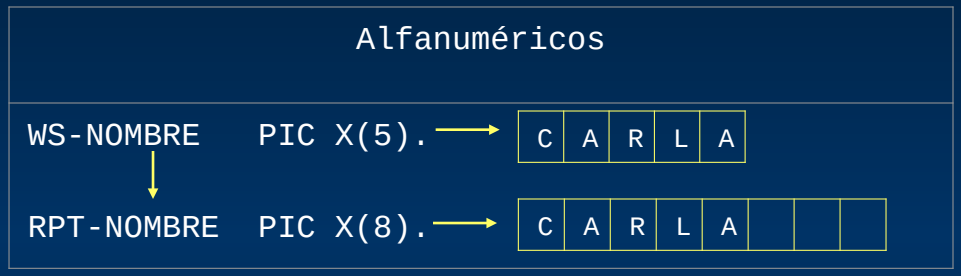
Tópico 330 - File Processing

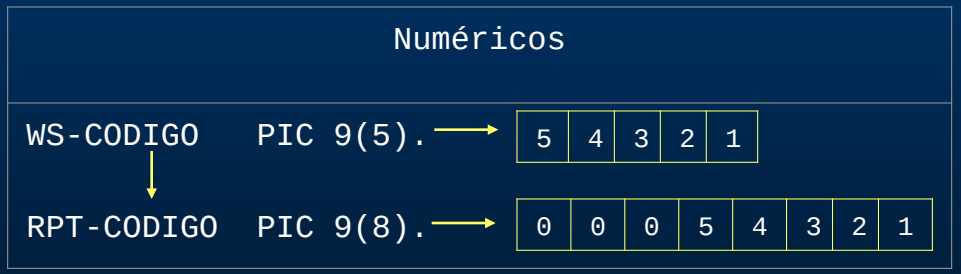
- Sentencia: **MOVE** (Cont.)
  - El truncamiento de los datos depende de la longitud del campo al que son informados:
    - Los campos alfanuméricos son alineados hacia la derecha
    - Los campos numéricos son alineados hacia la izquierda
  - Ejemplo1.
    - Cuando el campo receptor es más pequeño



Tópico 330 - File Processing

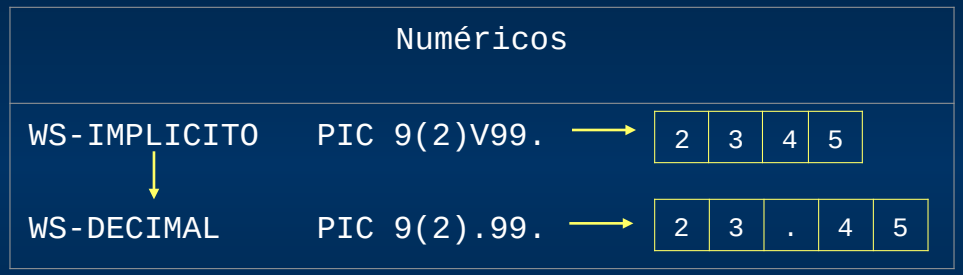
- Sentencia: **MOVE** (Cont.)
  - Ejemplo 2.
    - Cuando el campo receptor es más grande





Tópico 330 - File Processing

- Sentencia: **MOVE** (Cont.)
  - Ejemplo 3.
    - Cuando el campo que envía es un numérico con decimales implícitos y el que recibe es un numérico con dos decimales.



Tópico 330 - File Processing

- Sentencia: **READ**
  - Permite leer los registros de un archivo.
  - Su sintaxis:

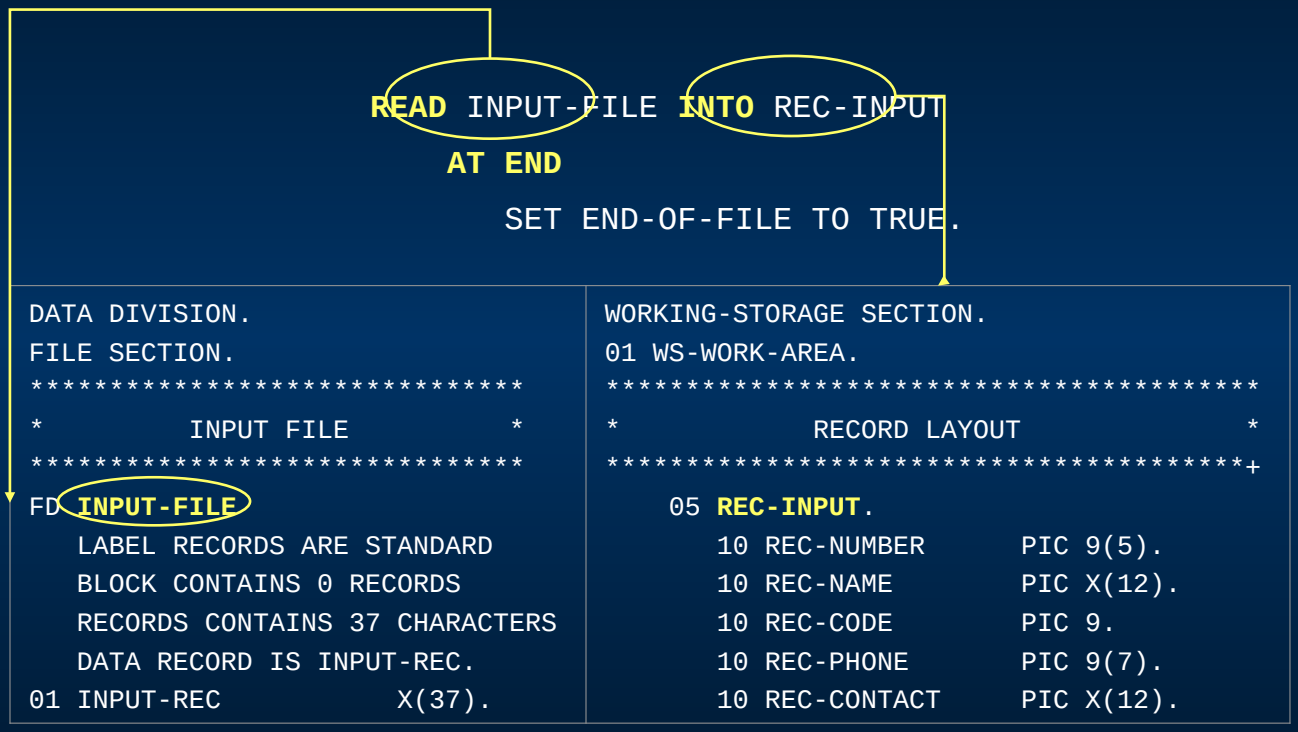
    READ *nombre-archivo* **INTO** *nombre-registro*

    AT END

    SET END-OF-FILE TO TRUE.
  - La sentencia **READ** lee el registro del archivo especificado cuyo nombre es el mismo que el asignado en la sección de **FD**.
  - Los datos del registro son transferidos a través de la sentencia **INTO** en la definición del registro que ha sido definido en la WS para el archivo.
  - Para controlar el fin de archivo en base a una condición es utilizada la cláusula **AT END**.

Tópico 330 - File Processing

- Sentencia: **READ** (Cont.)
  - Ejemplo :



Tópico 330 - File Processing

- Sentencia: **WRITE** (Cont.)
  - Esta sentencia permite escribir el contenido de la memoria a un dispositivo de salida (archivo, cinta, etc.).
  - Sintaxis:

WRITE *OUTPUT-REC*

  - A través de esta sentencia se puede escribir el contenido de un registro a un dispositivo de salida (archivo, cinta, etc.).
  - Su sintaxis:

WRITE *OUTPUT-REC* **FROM** *REC-DATA*

Tópico 330 - File Processing

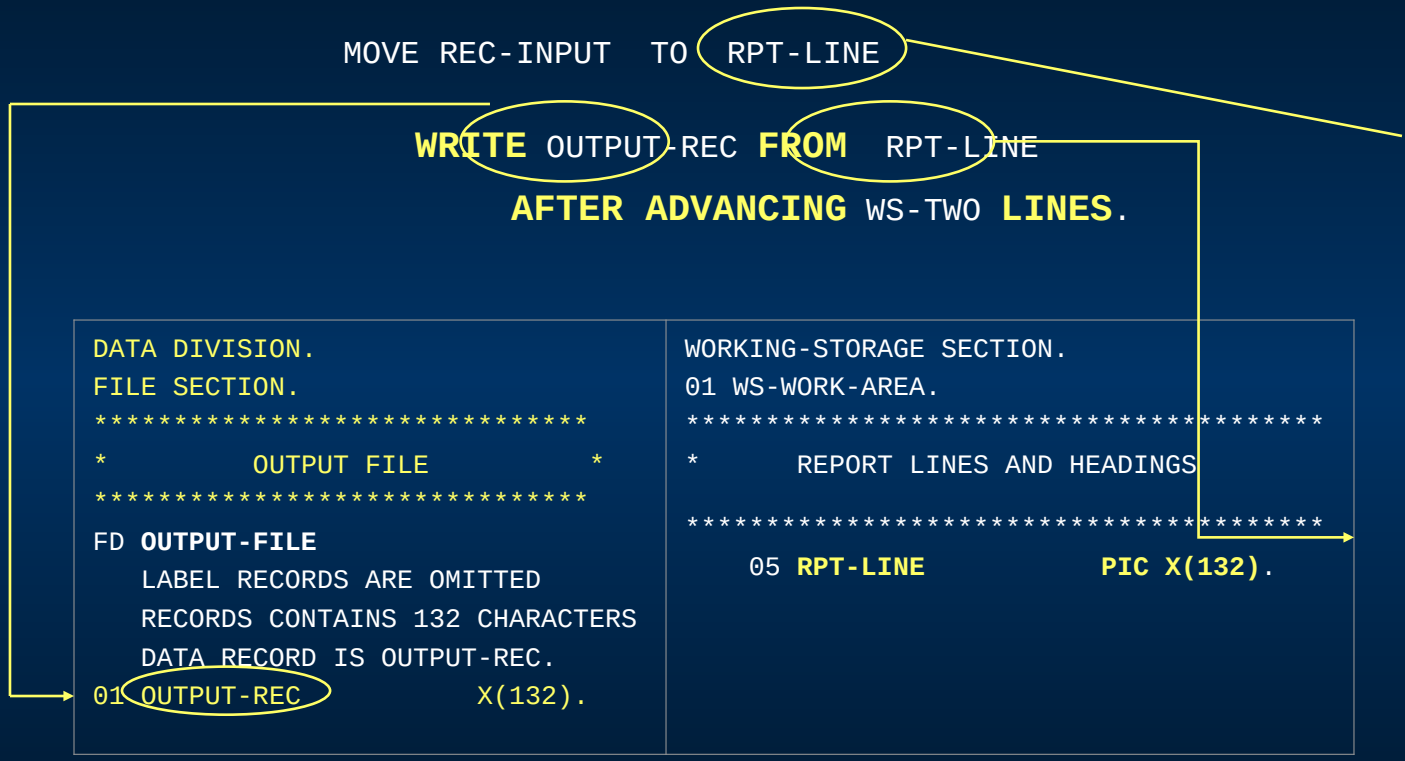
- Sentencia: **WRITE** (Cont.)
  - La cláusula **AFTER ADVANCING** es usada con el **WRITE** para el control de líneas de espaciamiento antes de imprimir la siguiente línea del reporte o comenzar a imprimir en una nueva página.
  - Su sintaxis:

WRITE *OUTPUT-REC* **FROM** *REC-DATA*

AFTER ADVANCING [*n* **LINES] [PAGE]**

Tópico 330 - File Processing

- Sentencia: **WRITE** (Cont.)



Tópico 330 - File Processing

- Sentencia: **CLOSE**
  - Esta sentencia permite cerrar los archivos de entrada y salida usados durante el proceso del programa.
  - Su sintaxis:

CLOSE *file-name1*
file-name2
file-name$_n$**.**

  - Ejemplo:

```
CLOSE      INPUT-FILE
           OUTPUT-FILE.
```

Módulo Básico

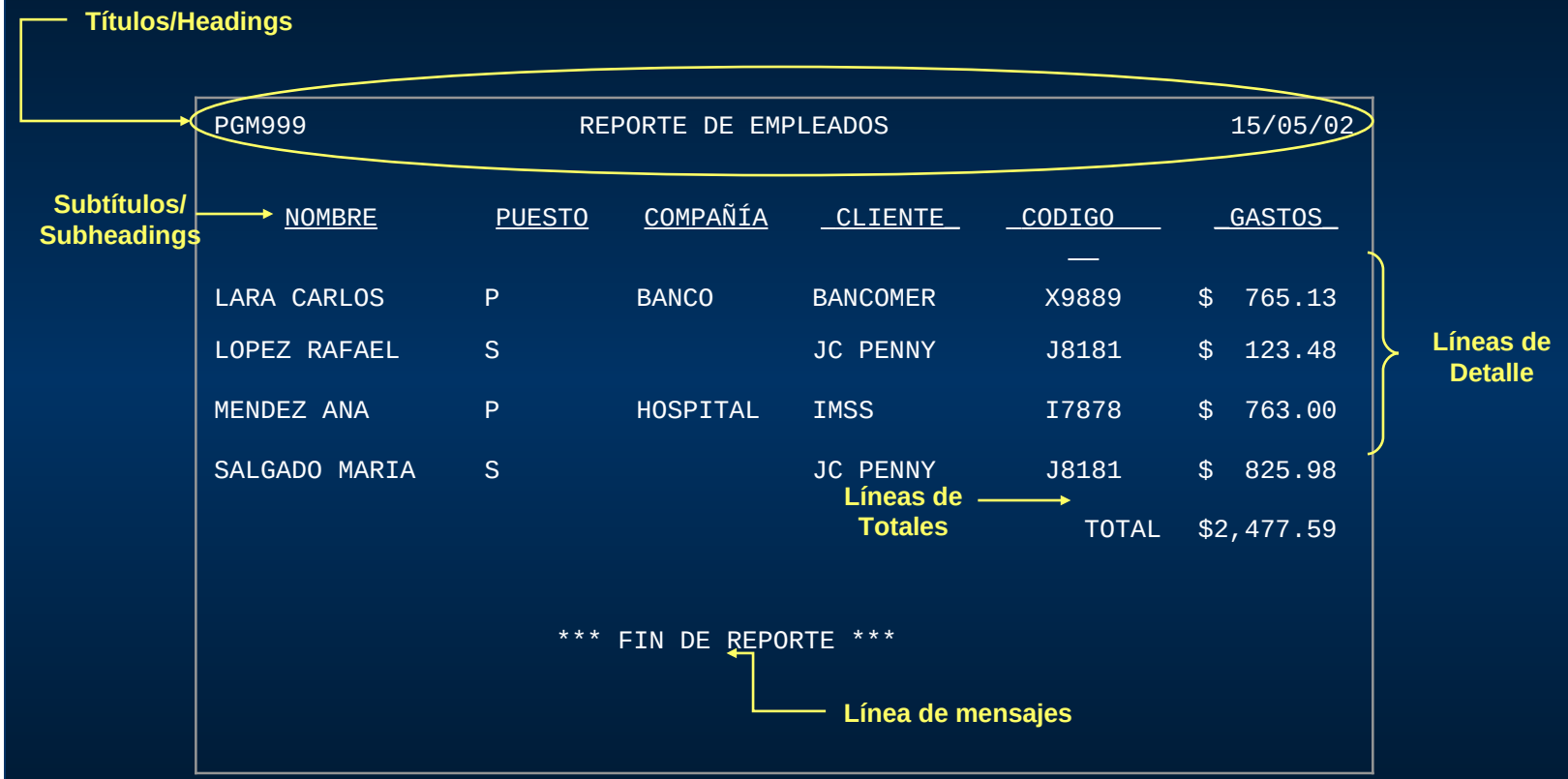
- Tópico 310 - Coding a COBOL Program
- Tópico 320 - Editing a Program
- Tópico 330 - File Processing
- **Tópico 340 - Formatting Data**
- Tópico 350 - COBOL Arithmetic and Program Logic
- Tópico 360 - Multimodular Programming and Copybooks
- Tópico 370 - Table Processing
- Tópico 380 - Galaxy Expense Processing System

Tópico 340 - Formatting Data

- OBJETIVOS.
  - Distinguir entre un dato variable y un dato constante.
  - Usar herramientas especiales de edición para hacer la información del reporte más fácil de leer y entender.
  - Transformar un dato no formateado en uno usado para la línea de un reporte.
  - Explicar el proceso del flujo de un programa que lee un registro a partir de un archivo, formatea los datos y genera un reporte.
  - Explicar el propósito de la función de la cláusula REDEFINES.

Tópico 340 - Formatting Data

- Tipos de líneas en un reporte



Tópico 340 - Formatting Data

- Tipos de información en un reporte.
  - Constante.
    - El valor del dato no cambia durante el tiempo de ejecución. Por ejemplo, títulos, subtítulos, etc.
    - La declaración de constantes está dada a través de las cláusulas **FILLER** y **VALUE** en la Working-Storage. Por ejemplo:

```
05  REPORT-LINES.
   10  RPT-HEADING1.
       15  FILLER              PIC X       VALUE SPACES.
       15  FILLER              PIC X(6)    VALUE 953940.
       15  FILLER              PIC X(38)   VALUE SPACES.
       15  FILLER              PIC X(17)
                          VALUE 'REPORTE DE PRUEBA'.
       15  FILLER              PIC X(70)   VALUE SPACES.
```

Tópico 340 - Formatting Data

- Tipos de información en un reporte.
  - Existen algunas cláusulas simbólicas que permiten asignar valores de acuerdo a la naturaleza del campo:
    - Asignar espacios a todas las posiciones de un campo alfanumérico: **SPACE, SPACES**
    - Asigna ceros a todas las posiciones de un campos numérico: **ZERO, ZEROS, ZEROES.**
  - La cláusula **VALUE ALL** permite indicar que el valor asignado ocupará todas las posiciones del campo.

```
05  REPORT-LINES.
      10  RPT-SUBHEADING1.
          15  FILLER              PIC X       VALUE SPACES.
                        15  FILLER              PIC X(15)   VALUE ALL '-'.
          15  FILLER              PIC 9(6)    VALUE ZEROES.
```

Tópico 340 - Formatting Data

- Tipos de información en un reporte.
  - Variable.
    - Cambia su información durante la ejecución del programa, por ejemplo, las líneas de detalle, los totales, etc.
    - Los datos variables no pueden ser asignados a un **FILLER**.
    - No se asigna generalmente un valor a los datos variables.

Tópico 340 - Formatting Data

- Edición de campos.

Tipo de edición	Campo fuente	Edicion	Campo sin editar	Campo editado
Edición de Decimales	PIC 9(999)V99	PIC 999.99	08457	084.57
Eliminación de ceros a la izquierda	PIC 99999.99	PIC ZZZZ9.99	00544.75	544.75
Comas	PIC 9(7)V99	PIC Z,ZZZ,ZZ9.99	000056788 473500991 000455679 055754907	567.88 4,735,009.00 4,556.79 557,549.07
Signo $ fijo	PIC 9(7)V99	PIC $Z,ZZZ,ZZ9.99	000056788 473500991 000455679 055754907	$ 567.88 $4,735,009.00 $ 4,556.79 $ 557,549.07

Tópico 340 - Formatting Data

- Edición de campos.

Tipo de edición	Campo fuente	Edicion	Campo sin editar	Campo editado
Signo $ flotante	PIC 9(7)V99	PIC $$,$$$,$$9.99	000056788 473500991 000455679 055754907	$567.88 $4,735,009.00 $4,556.79 $557,549.07
Adición de ceros	PIC 9(4)	PIC Z,ZZ9,000	0054	54,000
Inserción de blancos	PIC 9(6)	PIC 99B99B99	010191 060192	01 01 91 06 01 92
Adición de Signo “+”	PIC S9(4)	PIC +Z,ZZ9	+3456 +0005 -0032	+3,456 + 5 - 32
	PIC S9(4)	PIC Z,ZZ9+	+3456 +0005 -0032	3,456+ 5+ 32-

Tópico 340 - Formatting Data

- Edición de campos.

Tipo de edición	Campo fuente	Edicion	Campo sin editar	Campo editado
Adición de Signo “+” (Cont.)	PIC S9(4)	PIC ++,++9	+3456 +0005 -0032	+3,456 +5 -32
Adición de Signo “-”	PIC S9(4)	PIC -Z,ZZ9	+3456 +0005 -0032	3,456 5 - 32
	PIC S9(4)	PIC Z,ZZ9-	+3456 +0005 -0032	3,456 5 32-
	PIC S9(4)	PIC --,--9	+3456 +0005 -0032	3,456 5 -32

Tópico 340 - Formatting Data

- Edición de campos.

Tipo de edición	Campo fuente	Edicion	Campo sin editar	Campo editado
Adición de símbolos contables: CR - Positivo DB - Negativo	PIC S9(4)V99	PIC $Z,ZZ9.99CR	+164589 -164589	$1,645.89CR $1,645.89
	PIC S9(4)V99	PIC $Z,ZZ9.99DB	+164589 -164589	$1,645.89 $1,645.89DB
	PIC S9(4)V99	PIC $Z,ZZ9.99BCR	+164589 -164589	$1,645.89 CR $1,645.89
	PIC S9(4)V99	PIC $Z,ZZ9.99BDB	+164589 -164589	$1,645.89 $1,645.89 DB
Asteriscos	PIC S9(5)V99	PIC $**,***.**	0000000 0005380 0374750 1485500 1234567	$********* $****53.80 $*3,747.50 $*4,855.00 ERROR $*2,345.67 ERROR

Tópico 340 - Formatting Data

- Edición de campos.
  - Consideraciones importantes en la edición de datos:
    - En la eliminación de ceros y signo $.
      - Un campo no puede contener sólo ‘Z’ ó ‘$’
      - Las Z’s y $ no pueden ser codificados a la derecha de un 9 ó punto decimal.
      - El campo debe contener al menos un caracter 9 a la derecha del signo $.
    - Inserción de blancos
      - Pueden ser usados para campos numéricos y alfanuméricos
    - Signos ‘+’ y ‘-’
      - Cuando se coloca en el formato del campo el signo ‘+’, se imprime ‘+’ si el valor es positivo y ‘-’ si el valor es negativo.
      - Cuando se coloca en el formato del campo el signo ‘-’ éste sólo es impreso en valores negativos.
    - Signos CR y DB
      - El signo CR aparece sólo cuando el valor es positivo y el DB sólo cuando se trata de una cantidad negativa.

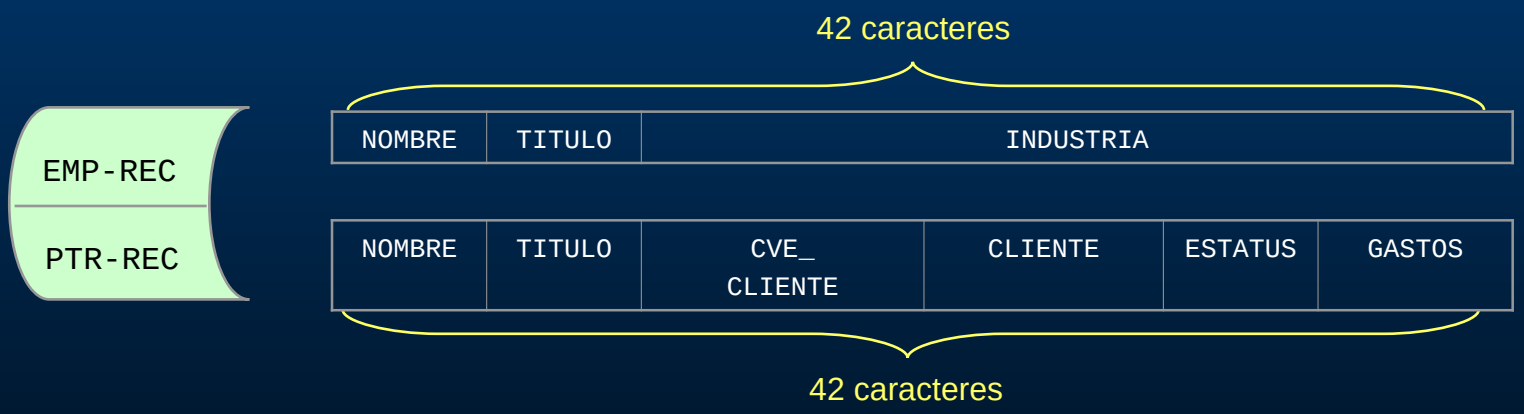
Tópico 340 - Formatting Data

- Edición de campos.
  - Consideraciones importantes en la edición de datos:
    - Asteriscos.
      - Pueden ser codificados a la derecha del punto decimal. Los asteriscos aparecerán aún cuando el valor sea cero.
      - Las comas son reemplazadas por asteriscos.
      - Pueden llegar a truncar un dígito del campo, ya que siempre imprimirá al menos un asterisco.

Tópico 340 - Formatting Data

- Cláusula: **REDEFINES**
  - Esta cláusula es usada para conservar el espacio de memoria, usando mas de una definición de registro para representar el mismo espacio de memoria.
  - Los campos pueden ser redefinidos por campos a su mismo nivel y no pueden ser utilizados en niveles 88.
  - No se les puede asignar cláusulas **VALUE**.
  - Su sintaxis:

campo-2 **REDEFINES** *campo-1*



Tópico 340 - Formatting Data

- Cláusula: **REDEFINES**

```
*************************************************************
*                        RECORD LAYOUTS                     *
*************************************************************
       10 EMP-REC.
          15 EMP-NOMBRE                 PIC X(12).
          15 EMP-TITULO                 PIC X.
          15 EMP-INDUSTRIA              PIC X(29).
       10 PTR-REC REDEFINES EMP-REC.
          15 PTR-NOMBRE                 PIC X(12).
          15 PTR-TITULO                 PIC X.
          15 PTR-CVE-CLIENTE            PIC X(8).
          15 PTR-CLIENTE                PIC X(10).
          15 PTR-ESTATUS                PIC X(6).
          15 PTR-GASTOS                 PIC 9(3)V99.
```

Tópico 340 - Formatting Data

- Creación del formato de un reporte.
  - Procesos principales para la creación de un reporte.
    - Lectura del archivo de entrada. En donde se lee el registro de entrada al campo ó campos que definen dicho registro.
    - Edición o formateo de los campos. Lo cual involucra mover cada campo del registro de entrada a su correspondiente campo en la línea de detalle del reporte.
    - Impresión del reporte. Involucra la escritura de la línea de detalle del reporte.

Módulo Básico

- Tópico 310 - Coding a COBOL Program
- Tópico 320 - Editing a Program
- Tópico 330 - File Processing
- Tópico 340 - Formatting Data
- **Tópico 350 - COBOL Arithmetic and Program Logic**
- Tópico 360 - Multimodular Programming and Copybooks
- Tópico 370 - Table Processing
- Tópico 380 - Galaxy Expense Processing System.

Tópico 350 - COBOL Arithmetic and Program Logic

- OBJETIVOS.
  - Describir qué es un campo compactado, cuándo y cómo usarlo.
  - Codificar correctamente campos numéricos (display, signados y compactados)
  - Describir el uso de los comandos: ADD, SUBSTRACT, MULTIPLY, DIVIDE, COMPUTE.
  - Definir y distinguir entre estructuras alternativas e iterativas
  - Codificar estructuras iterativas
  - Explicar la lógica de los IF anidados.

Tópico 350 - COBOL Arithmetic and Program Logic

- Campos numéricos.
  - Tipos de campos numéricos.
    - Display.
      - Impresos en pantallas, reportes.

Formato de campo	Ejemplo	Memoria Ocupada
PIC 99	25	2 bytes
PIC 99V	25	2 bytes
PIC 9(3)V99	32135	5 bytes

Tópico 350 - COBOL Arithmetic and Program Logic

- Campos numéricos.
  - Tipos de campos numéricos.
    - Signados.
      - Presentan el carácter '**S**' el cual indica la presencia de un signo de operación. Pueden tomar valores positivos o negativos.
      - Regularmente son usados para cálculos o procesos internos.

Formato de campo	Valor Almacenable	Memoria Ocupada
PIC S99	+25	2 bytes
PIC S99V	-25	2 bytes
PIC S9(3)V99	+321.35	5 bytes

Tópico 350 - COBOL Arithmetic and Program Logic

- Campos numéricos.
  - Tipos de campos numéricos.
    - Signados (Cont.)
      - Reservan un espacio en memoria para el signo, sin embargo, el campo sigue ocupando la misma cantidad de memoria.

Valor del signo en Hexadecimal		
Valor	Positivo	Negativo
0	{	}
1	A	J
2	B	K
3	C	L
4	D	M
5	E	N
6	F	O
7	G	P
8	H	Q
9	I	R

Por ejemplo :

Valor a guardar : -9385 Definición: **S**9(6) Representación: 00938N

Tópico 350 - COBOL Arithmetic and Program Logic

- Campos numéricos.
  - Tipos de campos numéricos.
    - Compactados.
      - Contienen dos dígitos por espacio de memoria o espacio de almacenamiento.
      - Son usados para cálculos aritméticos.

Valor a guardar : -12345.67 Definición: S9(5)V99 **COMP-3** Representación: 4V}

1	3	5	7
2	4	6	D

Total bytes de memoria = 4
(No. Dígitos + 1) / 2

Tópico 350 - COBOL Arithmetic and Program Logic

- Campos numéricos.
  - Tratamiento de datos entre campos numéricos.
    - El mover valores entre campos numéricos, implica que si uno de los campos es signado, el otro también lo sea, ya que de lo contrario se truncaría el signo, o bien se asumiría '+'.

PIC S9(3) MOVE CAMPO-ORIGEN -025	TO	PIC +999 CAMPO-RECEPTOR -025	CORRECTO
PIC S9(3) MOVE CAMPO-ORIGEN -025	TO	PIC 999 CAMPO-RECEPTOR 025	INCORRECTO
PIC 9(3) MOVE CAMPO-ORIGEN 025	TO	PIC +999 CAMPO-RECEPTOR +025	INCORRECTO

Tópico 350 - COBOL Arithmetic and Program Logic

- Campos numéricos.
  - Tratamiento de datos entre campos numéricos.
    - No se puede mover a nivel de grupo, un campo numérico a uno compactado directamente.

Definición en Working Storage.	Tratamiento en Procedure Division	Correcto /incorrecto
10 WS-DATE. 05 WS-AA PIC 99. 05 WS-MM PIC 99. 05 WS-DD PIC 99. 10 WS-DATE-COMP PIC 9(6) COMP-3.	MOVE WS-DATE TO WS-DATE-COMP	INCORRECTO

Tópico 350 - COBOL Arithmetic and Program Logic

- Campos numéricos.
  - Tratamiento de datos entre campos numéricos (Cont.)

Definición en Working Storage.	Tratamiento en Procedure Division	Correcto /incorrecto
10 WS-DATE. 05 WS-AA PIC 99. 05 WS-MM PIC 99. 05 WS-DD PIC 99. 10 WS-DATE-TEMP PIC 9(6). 10 WS-DATE-COMP PIC 9(6) COMP-3.	MOVE WS-DATE TO WS-DATE-TEMP MOVE WS-DATE-TEMP TO WS-DATE-COMP	CORRECTO
10 WS-DATE. 05 WS-AA PIC 99. 05 WS-MM PIC 99. 05 WS-DD PIC 99. 10 WS-DATE-RED REDEFINES WS-DATE PIC 9(6). 10 WS-DATE-COMP PIC 9(6) COMP-3.	MOVE WS-DATE-RED TO WS-DATE-COMP	CORRECTO

Tópico 350 - COBOL Arithmetic and Program Logic

- Verbos aritméticos.
  - **ADD**. Adición de valores.

    ADD *campo-1 campo-2 campo-N* **TO** *campo-z*

  - **SUBTRACT**. Sustracción de valores.

    SUBTRACT *campo-1 campo-2 campo-N* **FROM** *campo-z*

  - **MULTIPLY**. Multiplicación de valores. El resultado es almacenado en los campos especificados por la instrucción **BY.**

    MULTIPLY *campo-1* **BY** *factor-1 factor-2 factor-N*

MULTIPLY	A	**BY**	B	C
	2		3	10
Resultado	2		**6**	**20**

Tópico 350 - COBOL Arithmetic and Program Logic

- Verbos aritméticos.
  - **DIVIDE**. División de valores.

DIVIDE *dividendo* **BY** *divisor* **GIVING** *cociente*

DIVIDE	A	**BY**	B	**GIVING**	C
	10		2	⟶	**5**

DIVIDE *divisor* **INTO** *dividendo-1 dividendo-2*

DIVIDE	A	**INTO**	B
	2		10
	2		**5**

Tópico 350 - COBOL Arithmetic and Program Logic

- Verbos aritméticos.
  - **DIVIDE**. División de valores (cont).

DIVIDE *divisor* **INTO** *dividendo* **GIVING** *cociente* **REMAINDER** *residuo*

DIVIDE	A	INTO	B	GIVING	C	REMAINDER	D
	5		23	⟶	**4**		**3**

Tópico 350 - COBOL Arithmetic and Program Logic

- Verbos aritméticos.
  - **GIVING**. Asigna el valor de la operación al campo especificado.

    ADD *campo-1* **TO** *campo-z* **GIVING** *campo-suma*

    SUBTRACT *campo-1* **FROM** *campo-z* **GIVING** *campo-resta*

    MULTIPLY *campo-1* **BY** *factor-1* **GIVING** *campo-mult*

    DIVIDE *dividendo* **BY** *divisor* **GIVING** *cociente*

    DIVIDE *divisor* **INTO** *dividendo* **GIVING** *cociente* **REMAINDER** *residuo*

Tópico 350 - COBOL Arithmetic and Program Logic

- Verbos aritméticos.
  - **ROUNDED**. Redondea el valor del cálculo de la operación.
    - El valor es redondeado al siguiente superior cuando el valor truncado es mayor o igual a 5
    - El valor no es redondeado cuando el valor truncado es menor a 5

      ADD campo-1 **TO** campo-z **ROUNDED**

      SUBTRACT *campo-X* **FROM** *campo-z* **ROUNDED**

      MULTIPLY *campo-1* **BY** *fator* **GIVING** *campo-store* **ROUNDED**

Tópico 350 - COBOL Arithmetic and Program Logic

- Sentencia: **COMPUTE**
  - Esta sentencia permite realizar operaciones aritméticas sin necesidad de recurrir a los verbos aritméticos.
  - La sentencia **COMPUTE** utiliza los siguientes símbolos para especificar la operación:
    - \+ Suma
    - \- Resta
    - * Multiplicación
    - / División
    - ** Exponenciación
    - ( ) Paréntesis para jerarquía de operaciones

Tópico 350 - COBOL Arithmetic and Program Logic

- Sentencia: **COMPUTE**
  - Las prioridades de los signos para realizar las operaciones son:
    - ( ) Paréntesis
    - ** Exponenciación
    - * , / Multiplicación, División
    - +, - Suma, Resta
  - Las sentencias son evaluadas de izquierda a derecha dependiendo del operando.
  - Ejemplos.
    - COMPUTE A = A + B + C
    - COMPUTE A = (B*(C - D))

Tópico 350 - COBOL Arithmetic and Program Logic

- Sentencia: **COMPUTE**

USO DE COMPUTE

```
COMPUTE A =  (B * (C - D))
```

USO DE VERBOS

```
SUBTRACT D FROM C GIVING E

MULTIPLY B * E GIVING A
```

Tópico 350 - COBOL Arithmetic and Program Logic

- Estructuras lógicas
  - Existen dos tipos de estructuras lógicas:
    - ***Alternativas***. Es una sentencia condicional con una o más opciones. La opción realizada depende de la condición que resulte verdadera. Son codificadas con sentencias **IF.**
    - ***Iterativas***. Son sentencias condicionales que ocurren dentro de otra sentencia condicional.
  - Las estructuras alternativas e iterativas se clasifican en los siguientes tipos:
    - ***Simples***. Se valida una condición verdadera o falsa.
    - ***Compuestas***. Son aquellas que validan mas de una condición antes de tomar una opción. Cuando son usadas las cláusulas AND y OR en estas estructuras se evalúan todos los AND's y después los OR´s de izquierda a derecha**.**
    - ***Anidadas***. Son estructuras alternativas / iterativas que tienen que realizar dentro de otra estructura alternativas /iterativas.

Tópico 350 - COBOL Arithmetic and Program Logic

- Estructuras lógicas
  - Estructuras Alternativas.

Alternativas Simples	Alternativas Anidadas
IF *condición-1* *acción* **ELSE** *acción-alternativa* **END-IF.**	**IF** *condición-1* *acción-1* **IF** condición-2 *acción-2* **ELSE** *acción-3* **END-IF** **END-IF**.

Alternativas Compuestas
IF *condición-1* [**AND, OR**] *condición-2* *acción-1* **ELSE** *acción-alternativa* **END-IF**.

Tópico 350 - COBOL Arithmetic and Program Logic

- Estructuras lógicas
  - Estructuras Alternativas - Cláusulas.
    - La sentencia **NEXT SENTENCE** indica la continuación del proceso en la sentencia que se encuentra inmediatamente después del siguiente punto, en caso de que la condición no se cumpla.
    - Esta sentencia es opcional.

```
IF WS-A = CTE-CIEN
   ADD CTE-UNO  TO   ACC-TOTAL
ELSE
  NEXT SENTENCE
END-IF.

MOVE  CTE-CIEN TO WS-A
```

```
IF WS-A = CTE-CIEN
   ADD CTE-UNO  TO  ACC-TOTAL
END-IF.
MOVE  CTE-CIEN TO WS-A
```

Tópico 350 - COBOL Arithmetic and Program Logic

- Estructuras lógicas
  - Estructuras Alternativas - Cláusulas.
    - La sentencia **CONTINUE** indica la continuación del proceso en la siguiente sentencia que se encuentra. Esta sentencia es opcional.

```
IF WS-A = CTE-CIEN
   IF SW-PRIMERA-VEZ
      PERFORM 9900-ESCRIBE-CAB
          THRU 9900-ESCRIBE-CAB-EXIT
   ELSE
      CONTINUE
   END-IF
   PERFORM 9800-ESCRIBE-DETALLE
      THRU 9800-ESCRIBE-DETALLE-EXIT
ELSE
  MOVE ZEROES TO WS-B
END-IF.
MOVE  CTE-CIEN TO WS-A
```

Tópico 350 - COBOL Arithmetic and Program Logic

- Estructuras lógicas
  - Estructuras Alternativas - Condiciones.
  - Tipos de condiciones:
    - ***Condiciones de nombre***. Son aquellas que evalúan si un elemento es verdadero o falso (switches)
    - ***Condiciones de relación***. Se refiere a las condiciones que comparan un elemento con otro.

Afirmativas	Negativas
GREATER THAN	NOT GREATER THAN
LESS THAN	NOT LESS THAN
EQUAL TO	NOT EQUAL TO
GREATER THAN OR EQUAL TO	NOT GREATER THAN OR EQUAL TO
LESS THAN OR EQUAL TO	NOT LESS THAN OR EQUAL TO

Tópico 350 - COBOL Arithmetic and Program Logic

- Estructuras lógicas
  - Estructuras Alternativas - Condiciones.
    - ***Condiciones de clase***. Son aquellas usadas para datos que requieren ser validados antes de ser procesados. Se encuentran clasificadas en tres tipos:
      - *Numéricas*. Para caracteres numéricos.

        IF field **IS NUMERIC**
      - *Alfanuméricas*. Para caracteres alfanuméricos tales como letras, números y símbolos especiales, excepto algunos caracteres de control especiales.

        IF field **IS ALPHANUMERIC**
      - *Alfabéticas*. Para caracteres alfabéticos, espacios en blanco. Sólo puede ser usado si el campo fue declarado como alfabético.

        IF field **IS ALPHABETIC**

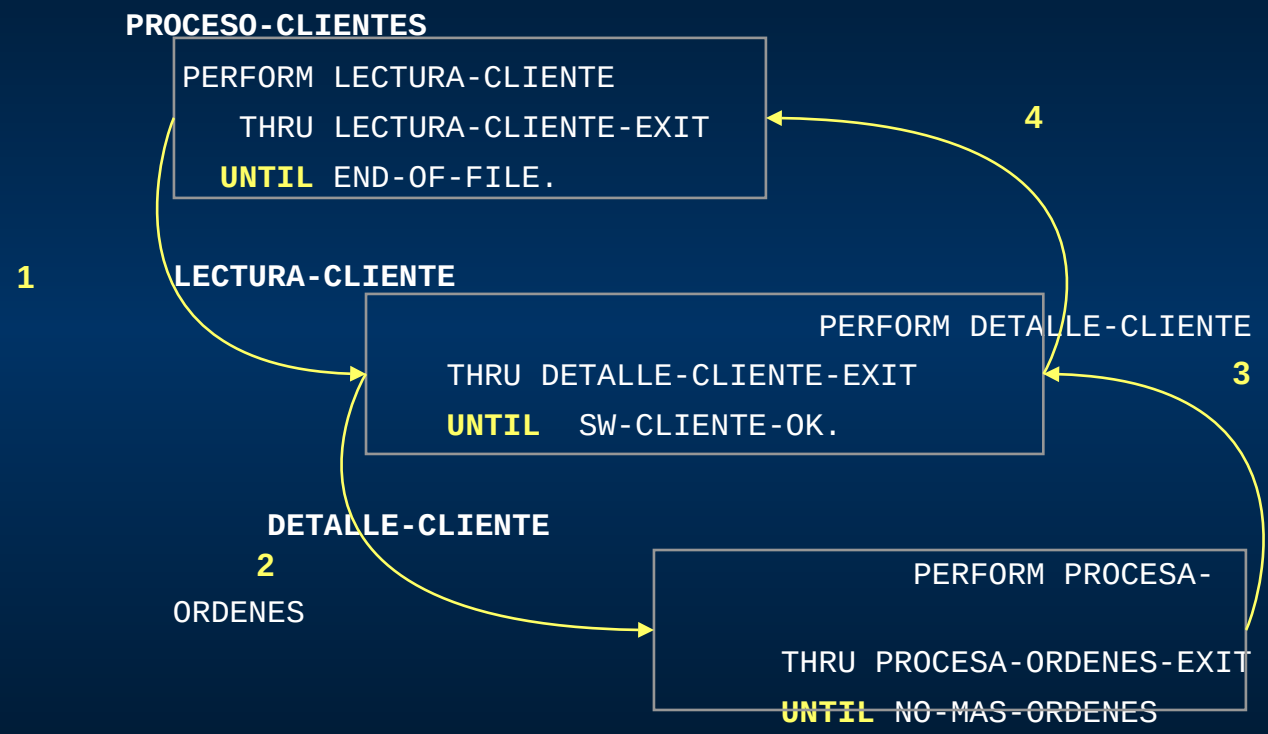
Tópico 350 - COBOL Arithmetic and Program Logic

- Estructuras lógicas
  - Estructuras Iterativas.

Iterativas Simples	Iterativas Compuestas
PERFORM *nombre-párrafo* THRU *nombre-párrafo-exit* **UNTIL** *condición* .	PERFORM *nombre-párrafo* THRU *nombre-párrafo-exit* **UNTIL** *condición-1* [AND,OR] *condición-2* .

Tópico 350 - COBOL Arithmetic and Program Logic

- Estructuras lógicas
  - Estructuras Iterativas.



Tópico 350 - COBOL Arithmetic and Program Logic

- Verbo **EVALUATE**
  - En el caso de requerir evaluar al mismo tiempo más de tres condiciones distintas, es posible utilizar la sentencia **EVALUATE**.

Sintaxis.

```
EVALUATE [identificador-1/ TRUE]
                WHEN condición-1
              acción-1
                WHEN condición-2
              acción-2
                WHEN condición-n
              acción-n
                WHEN OTHER
              acción-y
END-EVALUATE.
```

Módulo Básico

- Tópico 310 - Coding a COBOL Program
- Tópico 320 - Editing a Program
- Tópico 330 - File Processing
- Tópico 340 - Formatting Data
- Tópico 350 - COBOL Arithmetic and Program Logic
- **Tópico 360 - Multimodular Programming and Copybooks**
- Tópico 370 - Table Processing
- Tópico 380 - Galaxy Expense Processing System

Tópico 360 - Multimodular Programming and Copybooks

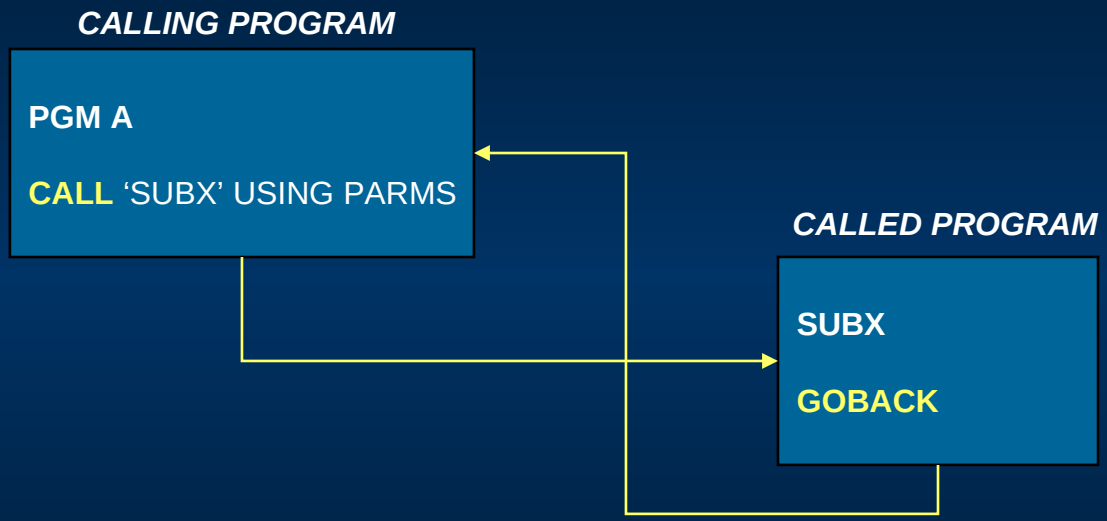
- OBJETIVOS.
  - Descripción y función de los programas multimodulares.
  - Diferencia entre un programa no modular y un programa modular.
  - Descripción de comandos de control entre programas.
  - Descripción y función de copybooks.

Tópico 360 - Multimodular Programming and Copybooks

- Programas Modulares.
  - Características:
    - Son programas que se encuentran divididos en funciones.
    - No realizan todo su proceso dentro del mismo programa, ya que hacen uso de uno o más programas para realizar ciertas funciones.
  - Tipos de programas modulares.
    - ***Programa principal (Calling program)***. Son aquellos que requieren de invocar a otro programa que realiza una función específica.
    - ***Subrutina o subprograma (Called program)***. Estos módulos requieren de otro programa para realizar una función específica y no pueden ser ejecutados por si mismos.

Tópico 360 - Multimodular Programming and Copybooks

- Programas Modulares.
  - Comunicación entre módulos.



Tópico 360 - Multimodular Programming and Copybooks

- Programas Modulares.
  - Comunicación entre módulos.
    - Programa principal (Calling module).
      - Se comunica a través de la siguiente sentencia:

        CALL ‘*nombre-modulo*’ **USING** *parámetro-comunicación1*
        parámetro-comunicación2
        parámetro-comunicación3.

      - Los parámetros de comunicación son declarados en la Working Storage.
      - Si la llamada a la rutina es en forma dinámica, el submodulo no es incluído durante la compilación del programa principal, con lo que no es necesario recompilar el programa principal por cada modificación realizada a la rutina, a menos que esto involucre algún cambio en los parámetros de comunicación.

Tópico 360 - Multimodular Programming and Copybooks

- Programas Modulares.
  - Comunicación entre módulos.
    - Subrutina (Called module).
      - Los parámetros de comunicación son declarados en la Linkage Section, en donde sólo se pasa la dirección de los datos y la información actual de estos.
      - Se comunica a través de la siguiente cláusula en la cabecera de la Procedure Division:

        PROCEDURE DIVISION **USING** *parámetro-comunicación1* *parámetro-comunicación2* *parámetro-comunicación3.*
      - Devuelve el control al programa principal con la cláusula **GOBACK.**

Tópico 360 - Multimodular Programming and Copybooks

- Programas Modulares.
  - Ventajas:
    - ***Complejidad y tamaño***. Las funciones pueden ser organizadas en pequeños módulos.
    - ***Manejo del proyecto***. Varios módulos pueden ser programados al mismo tiempo.
    - ***Mantenimiento***. Cuando funciones son codificadas en módulos separados, el código de una función en particular se encuentra en un solo lugar.
    - ***Consideraciones de lenguaje***. Debido a que los lenguajes son traducidos a lenguaje maquina cuando son compilados, cada modulo puede ser codificado en el lenguaje más apropiado.
    - ***El uso de memoria***. Un programa modular requiere menos memoria que un programa no modular.
  - Desventajas:
    - ***Cambios a los programas***. Se requiere conocer el tipo de llamada que se tiene a los módulos a partir de los programas principales, para no afectar los procesos que dependan de éstos.
    - ***Generación excesiva de módulos***. En caso de no conocer completamente las funciones que existen pueden duplicarse módulos.

Tópico 360 - Multimodular Programming and Copybooks

- Copybooks.
  - Un copybook es un pedazo de código, que puede sólo ser usado cuando es incluido como parte de un programa, conteniendo variables o funciones.
  - Se encuentran almacenados en diversas librerías.
  - No pueden ser ejecutados por ningún programa, ni por si mismos.
  - Puede ser usado en la Working Storage o Procedure Division.
  - Su sintaxis:

    COPY *nombre-copybook*.
  - Sus ventajas:
    - Reducen el tiempo de codificación de programación
    - Fácil mantenimiento ya que sólo existe una sola copia del código

Módulo Básico

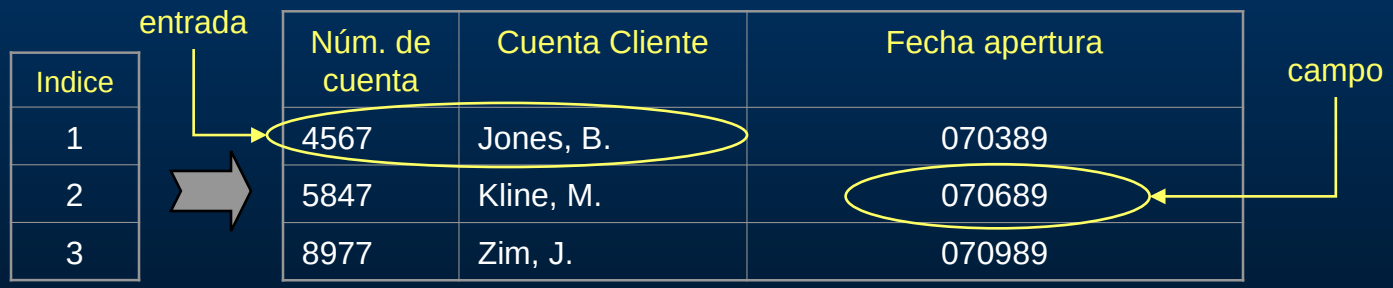
- Tópico 310 - Coding a COBOL Program
- Tópico 320 - Editing a Program
- Tópico 330 - File Processing
- Tópico 340 - Formatting Data
- Tópico 350 - COBOL Arithmetic and Program Logic
- Tópico 360 - Multimodular Programming and Copybooks
- **Tópico 370 - Table Processing**
- Tópico 380 - Galaxy Expense Processing System

Tópico 370 - Table Processing

- OBJETIVOS.
  - Definir una tabla interna y externa en la Working Storage Section.
  - Codificar la lógica de la carga de una tabla externa.
  - Codificar la lógica para realizar una búsqueda serial y una búsqueda binaria en una tabla.

Tópico 370 - Table Processing

- Definición de Tablas
  - Una tabla es un conjunto de valores almacenados en una posición de almacenamiento consecutivo y con un nombre de dato asignado.
  - Componentes de una tabla:
    - ***Entrada***. Consiste de uno ó mas campos (renglón). Todas las entradas tienen el mismo nombre y son accedidas por su posición relativa en la tabla.
    - ***Indice***. El cual mantiene la posición relativa de la entrada que está siendo accedida.



Tópico 370 - Table Processing

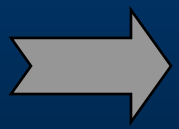
- Definición de Tablas
  - Tablas internas.
    - Son almacenadas en la memoria de los programas.
    - Son codificadas dentro de los programas, cuando éstas contienen pocos datos que raramente cambiarán.
    - Los elementos necesarios para definir una tabla interna son:
      - Los datos que contendrá la tabla.
      - Definición de la tabla.
    - Los verbos utilizados para la definición de una tabla son:
      - **REDEFINES**
      - **OCCURS**
      - **INDEXED BY**

Tópico 370 - Table Processing

- Definición de Tablas
  - Tablas internas (Cont.)

Tabla de datos

Mes	Día
Enero	31
Febrero	28
Marzo	31
Abril	30
Mayo	31
Junio	30
Julio	31
Agosto	31
Septiembre	30
Octubre	31
Noviembre	30
Diciembre	31



Definición de la tabla en Working Storage

```
05 MESES.
   10 FILLER    PIC X(12) VALUE 'ENERO     31'.
   10 FILLER    PIC X(12) VALUE 'FEBRERO   28'.
   10 FILLER    PIC X(12) VALUE 'MARZO     31'.
   10 FILLER    PIC X(12) VALUE 'ABRIL     30'.
   10 FILLER    PIC X(12) VALUE 'MAYO      31'.
   10 FILLER    PIC X(12) VALUE 'JUNIO     30'.
   10 FILLER    PIC X(12) VALUE 'JULIO     31'.
   10 FILLER    PIC X(12) VALUE 'AGOSTO    31'.
   10 FILLER    PIC X(12) VALUE 'SEPTIEMBRE30'.
   10 FILLER    PIC X(12) VALUE 'OCTUBRE   31'.
   10 FILLER    PIC X(12) VALUE 'NOVIEMBRE 31'.
   10 FILLER    PIC X(12) VALUE 'DICIEMBRE 31'.
```

Tópico 370 - Table Processing

- Definición de Tablas
  - Tablas internas (Cont.)

```
05 TABLA-MESES     REDEFINES  MESES.
   10 MESES-ENTRADA      OCCURS  12 TIMES
                         INDEXED BY MESES-INDEX.
      15 TBL-MES            PIC X(10).
      15 TBL-DIA            PIC X(02).
```

- Los índices son campos numéricos, no pueden ser display ni compactados. Sólo pueden ser usados con sentencias SET ó PERFORM.
- Inicializar un índice:

  SET índice **TO** 1.

- Incrementar un índice:

  SET índice **UP BY** 1.

Tópico 370 - Table Processing

- Definición de Tablas
  - Tablas externas.
    - Son almacenadas fuera de los programas en discos y en otros medios externos.
    - Son almacenadas externamente cuando:
      - Son muy grandes.
      - Contendrán información que cambia frecuentemente .
      - Serán accedidas por otros programas.
    - Cuando las tablas externas son usadas frecuentemente por el programa son cargadas en la Working Storage.

Tópico 370 - Table Processing

- Definición de Tablas
  - Tablas externas.

Núm. de cuenta	Cuenta Cliente	Fecha apertura
4567	Jones, B.	070389
5847	Kline, M.	070689
8977	Zim, J.	070989

```
05 TABLA-CUENTAS.
   10 CUENTAS  OCCURS  250 TIMES
              INDEXED BY CUENTAS-INDEX.
      15 NUMERO                  PIC 9(04).
      15 NUM-CLIENTE    PIC X(15).
      15 FECHA-APERTURA        PIC 9(06)
```

Tópico 370 - Table Processing

- Definición de Tablas
  - Lógica de carga de una tabla externa.

Housekeeping
- Abrir archivo de entrada de los datos de la tabla
- Inicializar switch de no fin de archivo
- Incializar el índice a 1
- Leer archivo de entrada

- Cargar tabla

(Until end-of-file)
- Mover datos del archivo de entrada al campo de la tabla(índice)
- Incrementar el índice en 1
- Leer el siguiente registro del archivo

Tópico 370 - Table Processing

- Definición de Tablas
  - Beneficio del uso de tablas.
    - El uso de tablas internas evita el acceso a datos desde un dispositivo externo, haciéndolo más eficiente en el caso de tratarse de una cantidad de datos pequeña.
    - El uso de tablas externas permite que más de un programa pueda usarlas, reduciendo el tiempo de desarrollo y el esfuerzo de mantenimiento, además de asegurar la integridad de la información que se maneja.

Tópico 370 - Table Processing

- Métodos de búsquedas en tablas
  - Búsqueda Serial.
    - Cada renglón de la tabla es accedido en forma secuencial hasta que el registro cumple con la condición especificada o se llaga al fin de la tabla.
    - Los datos de la tabla pueden encontrarse en cualquier orden.
    - Al realizar una búsqueda secuencial, el índice debe comenzar en 1.

```
SET index   TO  1.
SEARCH entrada-tabla
AT END
        acción-1
WHEN condición
        acción-2
END-SEARCH.
```

Tópico 370 - Table Processing

- Métodos de búsquedas en tablas
  - Búsqueda Binaria.
    - Este método de búsqueda es utilizado en tablas muy grandes y sólo cuando están ordenadas por un campo llave en forma ascendente, descendente o llevan una secuencia especifica.
    - Los argumentos son comparados con el valor del campo llave localizado a la mitad de la tabla. Durante éste proceso se va descartando la primera o segunda mitad de la tabla dependiendo del valor del argumento, hasta encontrar o finalizar la lectura.
    - Reglas:
      - La tabla debe estar ordenada de acuerdo al campo llave en forma ascendente o descendente.
      - El valor del campo llave debe ser un valor numérico que permita usar un criterio de ascendente o descendente.
      - La división de la tabla se hace de acuerdo a los renglones que contengan información.

Tópico 370 - Table Processing

- Métodos de búsquedas en tablas
  - Búsqueda Binaria (Cont.)
    - La definición de la tabla tiene una cláusula extra:

[ASCENDING, DESCENDING] **KEY IS** *campo-llave*

```
05 TABLA-CUENTAS.
   10 CUENTAS  OCCURS  250 TIMES
              ASCENDING KEY IS NUMERO
              INDEXED BY CUENTAS-INDEX.
      15 NUMERO                  PIC 9(04).
      15 NUM-CLIENTE    PIC X(15).
      15 FECHA-APERTURA        PIC 9(06)
```

Tópico 370 - Table Processing

- Métodos de búsquedas en tablas
  - Búsqueda Binaria (Cont.)
    - En éste tipo de búsqueda no es necesario iniciar el índice en 1, ya que la sentencia determina el valor de éste.

```
SEARCH ALL entrada-tabla
AT END
        acción-1
WHEN condición
        acción-2
END-SEARCH.
```

Tópico 370 - Table Processing

- Procesamiento de Tablas.
  - Para realizar la lectura secuencial de una tabla sin utilizar **SEARCH,** el proceso lógico sería:

Lectura de tabla

- Iniciar índice en 1.
- Leer tabla (UNTIL índice > total-occurs)
  - Mover *campo1-tabla*(*índice*) a *rpt-campo1*
  - Mover *campo2-tabla*(*índice*) a *rpt-campo2*
  - Mover *campon-tabla*(*índice*) a *rpt-campon*
  - Escribir registro.
  - Incrementar índice en 1.
  - Inicializar campos rpt.

Tópico 370 - Table Processing

- Procesamiento de Tablas - Tablas bidimensionales.

REGION	TRIMESTRE			
	1	2	3	4
1				
2				
3				
4				

```
WORKING-STORAGE SECTION..
01  REGION                                     PIC 9(1).
01  TRIMESTRE                                  PIC 9(1).

01 TABLA-VENTAS.
     05  DATOS-TRIMESTRE    OCCURS 4 TIMES.
        10 DATOS-REGION          OCCURS 4 TIMES.
             15 IMPORTE                        PIC 9(5).

PROCEDURE DIVISION.
....
....
       PERFORM 1100-IMPRIMIR-TABLA
          THRU 1100-IMPRIMIR-TABLA-EXIT
       VARYING TRIMESTRE FROM 1 BY 1
          UNTIL TRIMESTRE GREATER THAN CTE-FOUR
...
...
1100-IMPRIMIR-TABLA.
     PERFORM 1120-MOVER-DATOS
        THRU 1120-MOVER-DATOS
      VARYNG REGION FROM 1 BY 1
       UNTIL REGION GREATER THAN CTE-FOUR

     MOVE IMPORTE(TRIMESTRE,REGION) TO SALIDA.
```

Tópico 370 - Table Processing

- Procesamiento de Tablas - Tablas bidimensionales.
  - Reglas de uso para la cláusula **OCCURS:**
    - La cláusula OCCURS no puede usarse con elementos a nivel 77.
    - El máximo de ocurrencias permitido hasta el momento son 7 dimensiones.
    - Los subíndices pueden ser constantes enteras o variables enteras. Sus valores deben ser positivos; no pueden ser cero o negativo.
    - Los subíndices deben estar dentro de paréntesis y deben estar separados por comas.

Módulo Básico

- Tópico 310 - Coding a COBOL Program
- Tópico 320 - Editing a Program
- Tópico 330 - File Processing
- Tópico 340 - Formatting Data
- Tópico 350 - COBOL Arithmetic and Program Logic
- Tópico 360 - Multimodular Programming and Copybooks
- Tópico 370 - Table Processing
- **Tópico 380 - Galaxy Expense Processing System**

Tópico 380 - Galaxy Expense Processing System

- OBJETIVOS.
  - Aprender la funcionalidad de un programa batch.
  - Identificar la entradas y salidas de un programa en COBOL.
  - Codificación de programas batch.
  - Manejo de archivos indexados.

Tópico 380 - Galaxy Expense Processing System

- Archivos Indexados.
  - Se encuentran organizados por un área de índices y otra de datos, haciendo referencia a un campo clave.
  - El acceso a los registros se realiza localizando en el índice la posición de memoria, para después acceder a ésta y encontrar el registro correspondiente, sin necesidad de recorrer toda el área de datos.

SELECT *nombre-interno-archivo*
ASSIGN TO *nombre-externo*
ORGANIZATION IS INDEXED
ACCESS MODE IS [RANDOM, SEQUENTIAL]
RECORD KEY IS *llave-acceso*.

  - El almacenamiento de los archivos indexados es realizado sólo en discos.

Tópico 380 - Galaxy Expense Processing System

- Archivos Indexados.
  - El nombre del campo llave debe estar definido en el registro de FD.

```
FD   nombre-interno-archivo
     LABEL RECORDS ARE STANDARD
     BLOCK CONTAINS 0 RECORDS
     RECORDS CONTAINS n CHARACTERS.
01 nombre-lógico-registro.
     05   llave-acceso          PIC X(##).
     05   FILLER                PIC X(##).
```

  - El total de caracteres '*n*' debe ser igual a la suma de la longitud de los campos definidos en el nombre lógico del registro.

Tópico 380 - Galaxy Expense Processing System

- Archivos Indexados.
  - Modo de acceso **SECUENCIAL**.
    - Mientras que los archivos secuenciales son leídos de acuerdo al orden en que fueron escritos, los archivos indexados son leídos en el orden determinado por el campo clave, primero es leído el campo clave más pequeño.
    - En el modo de acceso **SECUENCIAL** sólo se permite apertura de **INPUT**, **OUTPUT** .
    - La apertura de **INPUT** en modo de acceso **SECUENCIAL** de archivos indexados se realiza de la siguiente forma:

READ *nombre-archivo-interno* **INTO** *nombre-lógico-registro*

AT END

SET END-OF-FILE **TO TRUE**.

El final del archivo se detecta en el índice y no en los datos.

Tópico 380 - Galaxy Expense Processing System

- Archivos Indexados.
  - Modo de acceso **SECUENCIAL** *(Cont.)*.
    - En caso de querer posicionarse a partir de un valor de índice específico se utiliza la sentencia **START**.

      START *nombre-archivo-interno* **KEY** [=,>,<,>=] *dato-clave*

      INVALID KEY

      *acción***.**

    - La apertura de **OUTPUT** en modo de acceso **SECUENCIAL** se realiza:

      WRITE *nombre-archivo-interno* **FROM** *detalle-datos*

      INVALID KEY ← Evita la adición de registros duplicados

      *acción***.**

Tópico 380 - Galaxy Expense Processing System

- Archivos Indexados.
  - Modo de acceso **RANDOM.**
    - Los archivos indexados con acceso **RANDOM** permiten recuperar cualquier registro indicando sólo el valor de la clave a buscar.
    - Sólo se permite de apertura **INPUT, OUTPUT, I-O** .
    - La apertura de **INPUT** sólo permite la lectura de los registros, con lo que en modo de acceso **RANDOM** de archivos indexados se realiza:

MOVE *valor-campo-llave* **TO** *llave-acceso.*

READ *nombre-archivo-interno* **INTO** *nombre-lógico-registro*

AT END

SET END-OF-FILE **TO TRUE**.

Tópico 380 - Galaxy Expense Processing System

- Archivos Indexados.
  - Modo de acceso **RANDOM** *(Cont.)***.**
    - La adición en apertura de **I-O** se realiza:

MOVE *valor-campo-llave* **TO** *llave-acceso.*

WRITE *nombre-archivo-interno* **FROM** *detalle-datos*

INVALID KEY

*acción***.**

Al adicionar el registro, se realiza una búsqueda automática sobre el índice, para localizar la posición donde se deba añadir el nuevo registro, comprobando adicionalmente que la clave no se encuentre duplicada.

Tópico 380 - Galaxy Expense Processing System

- Archivos Indexados.
  - Modo de acceso **RANDOM** *(Cont.)*.
    - La modificación en apertura **I-O** se realiza:

MOVE *valor-campo-llave* **TO** *llave-acceso.*
REWRITE *nombre-archivo-interno* **FROM** *detalle-datos*
INVALID KEY
acción.

    - El borrado en apertura **I-O** se realiza:

MOVE *valor-campo-llave* **TO** *llave-acceso.*
DELETE *nombre-archivo-interno*
INVALID KEY
acción.

Tópico 380 - Galaxy Expense Processing System

- Archivos Indexados.
  - Modo de acceso **RANDOM** *(Cont.)***.**
    - La lectura en apertura **I-O** se realiza:

MOVE *valor-campo-llave* **TO** *llave-acceso.*

READ *nombre-archivo* [**INTO** *layout-registro]*

INVALID KEY

*acción***.**

Tópico 380 - Galaxy Expense Processing System

- Codificación de un programa batch que realizará lo siguiente:
  - Lee un archivo de gastos (Expense Transaction) en el que se valida lo siguiente:
    - La existencia de cada uno de estos registros en el archivo de registro de gastos (Registrant Expense).
    - El formato de cada uno de los campos que integra el registro del archivo de gastos.
    - La existencia del código de gasto y número de registro en los archivos Expense Decode y Registrant Expense respectivamente.
  - Se genera de salida un archivo de errores y se modifica el archivo Registrant Expense con la información del archivo expense Transaction.

Módulo Específico

Módulo Específico

- **Tópico 410 - Data Storage & Data Base Management System**
- Tópico 420 - Accesing Data in a Data Base
- Tópico 430 - Accessing Data from within a Program.
- Estándares de programación SQL.

Tópico 410 - Data Storage & Data Base Management System

- OBJETIVOS.
  - Definir una Base de Datos.
  - Identificar los diferentes tipos de Bases de Datos.
  - Definir un Sistema Manejador de Base de Datos.
  - Describir funciones, ventajas y desventajas de un Manejador de Base de Datos.
  - Describir las ventajas de una Base de datos relacional.
  - Explicar las características básicas del Sistema Manejador de Bases de Datos DB2.

Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Conceptos.
    - **Base de Datos**. Colección de archivos que contienen datos.
    - **Manejador de Base de datos (DBMS).** Es un grupo de programas especialmente escritos que pueden localizar datos existentes en una Base de datos.

Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Organización.
    - Física.
      - Los datos deben ser almacenados en una posición especifica de la Base de Datos.
      - Los datos tienen un tamaño asociado, el cual debe ser tomado en cuenta cuando son almacenados.

Ejemplo:

Datos	Libros de una biblioteca
Posición de los datos dentro de la base da datos	Piso o ubicación donde se encuentra el libro.
Tamaño de los datos (número de bytes)	Características físicas del libro

Tópico 410 - Data Storage & Data Base Management System

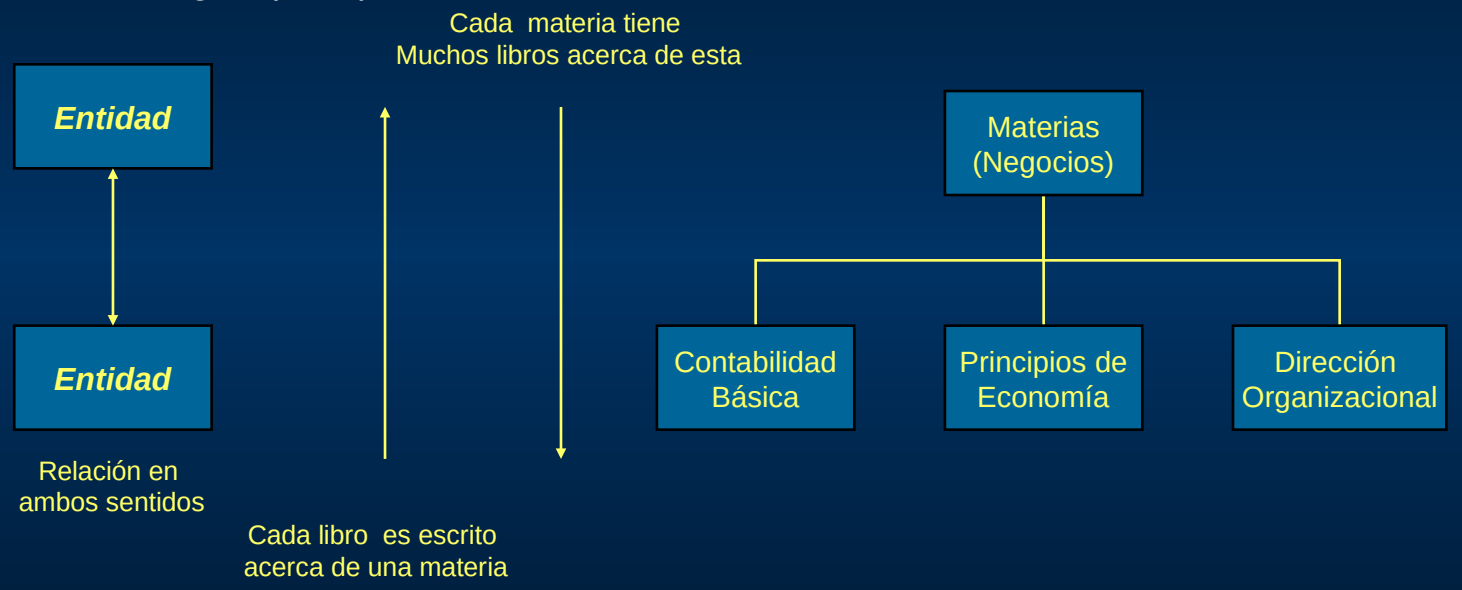
- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Organización.
    - Física (*cont.*)
      - El almacenamiento de las bases de datos es externa a la Working Storage de un programa COBOL.
      - Las principales características del almacenamiento externo de datos son:
        - ✓ Pueden ser accedidos rápidamente.
        - ✓ El acceso a los datos puede ser limitado a los datos que sólo quieran consultarse.
        - ✓ Acceder a ciertos datos que puedan ser restringidos a los usuarios.
        - ✓ Requiere menos espacio de memoria en la Working Storage.
        - ✓ La información se encuentra disponible para múltiples usuarios.

Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Organización.
    - Lógica.
      - Se refiere a las relaciones lógicas entre los datos que están siendo almacenados en la Base de datos, sin necesidad de conocer su actual posición en memoria o física en la Base de datos.
      - Elementos de la Organización lógica:
        - ***Entidad***. Término usado para refererirse a un objeto en particular que es representado en una Base de datos.
        - ***Relación***. Es la correspondencia que existe entre las Bases de Datos.
        - ***Ocurrencias***. Mientras una entidad se refiere a un objeto almacenado en una base de datos, las ocurrencias se refieren a una entidad especifica.

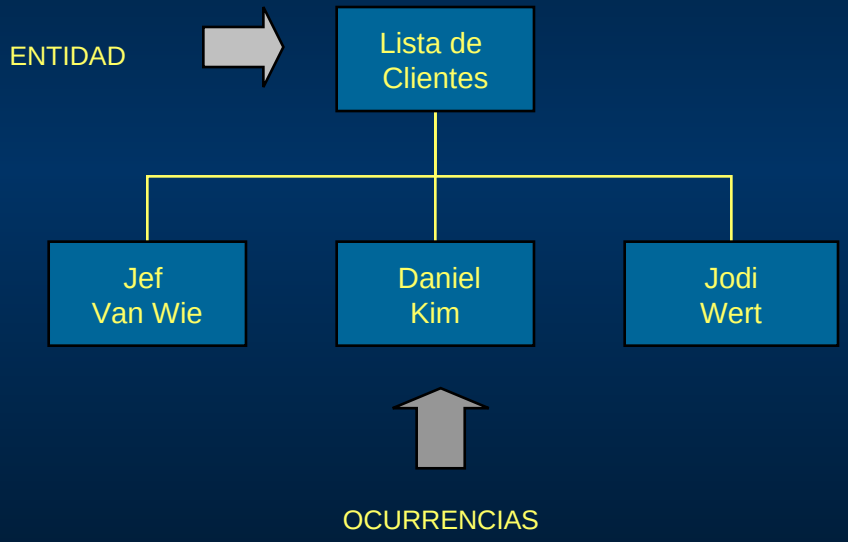
Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Organización.
    - Lógica *(cont.)*



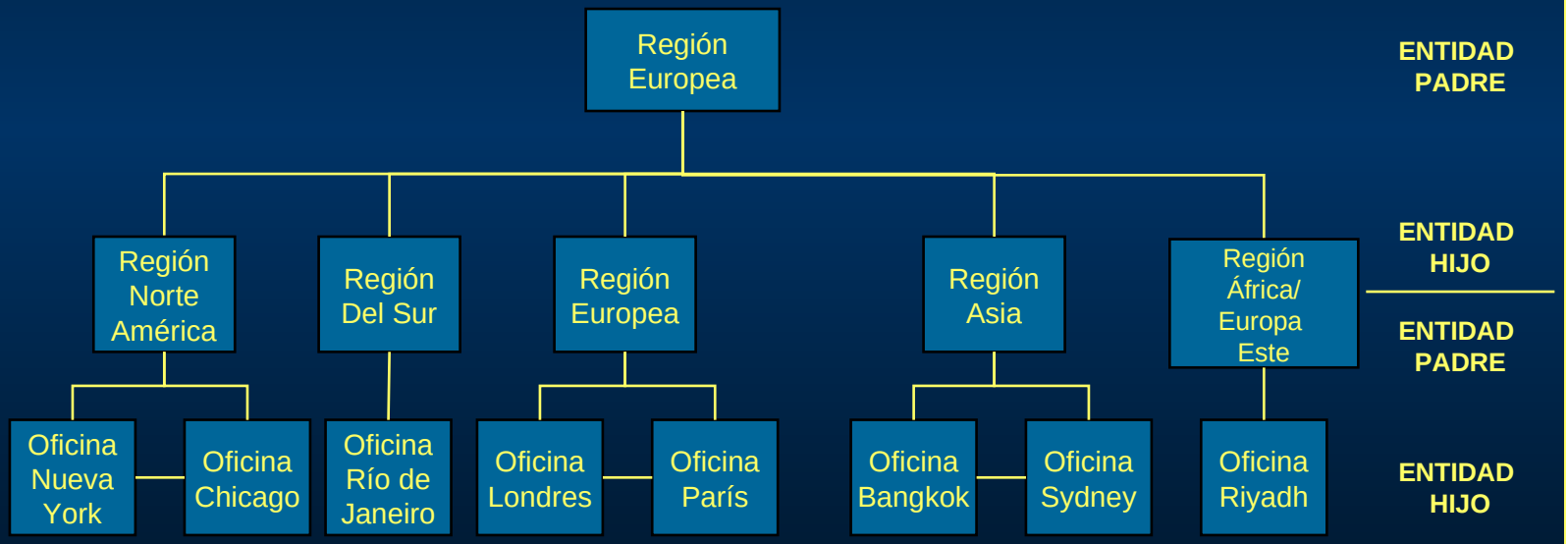
Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Organización.
    - Lógica *(cont.)*



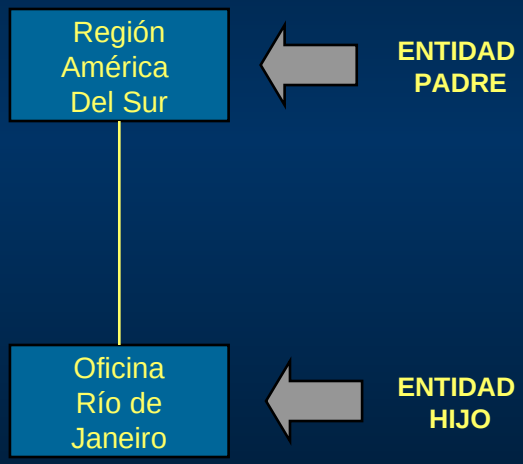
Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Organización.
    - Lógica *(cont.)*
      - Representación de una entidad que pertenece a otra entidad.



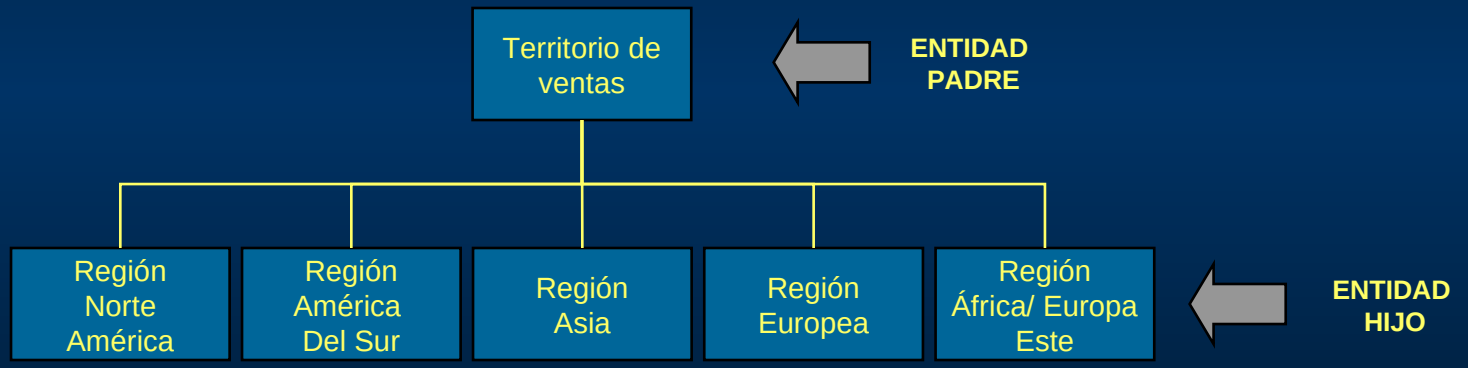
Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Relaciones entre entidades.
    - **Uno a uno**. Significa que a un padre le pertenece un hijo.



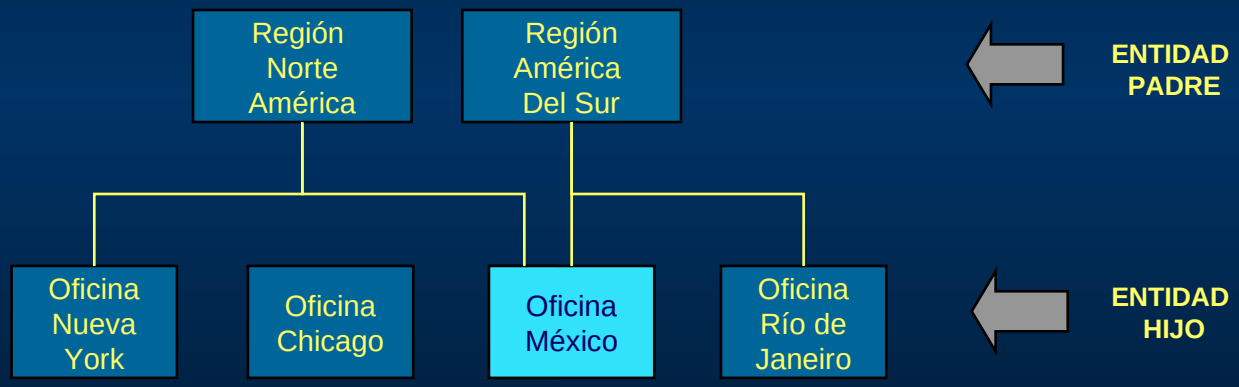
Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Relaciones entre entidades.
    - **Uno a muchos**. Esta relación existe cuando a un padre le pertenece más de un hijo, y cada hijo tiene sólo un padre.



Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Relaciones entre entidades.
    - **Muchos a muchos**. Esta relación existe cuando a un padre le pertenecen más de un hijo, y cada hijo le corresponden múltiples padres.



Tópico 410 - Data Storage & Data Base Management System

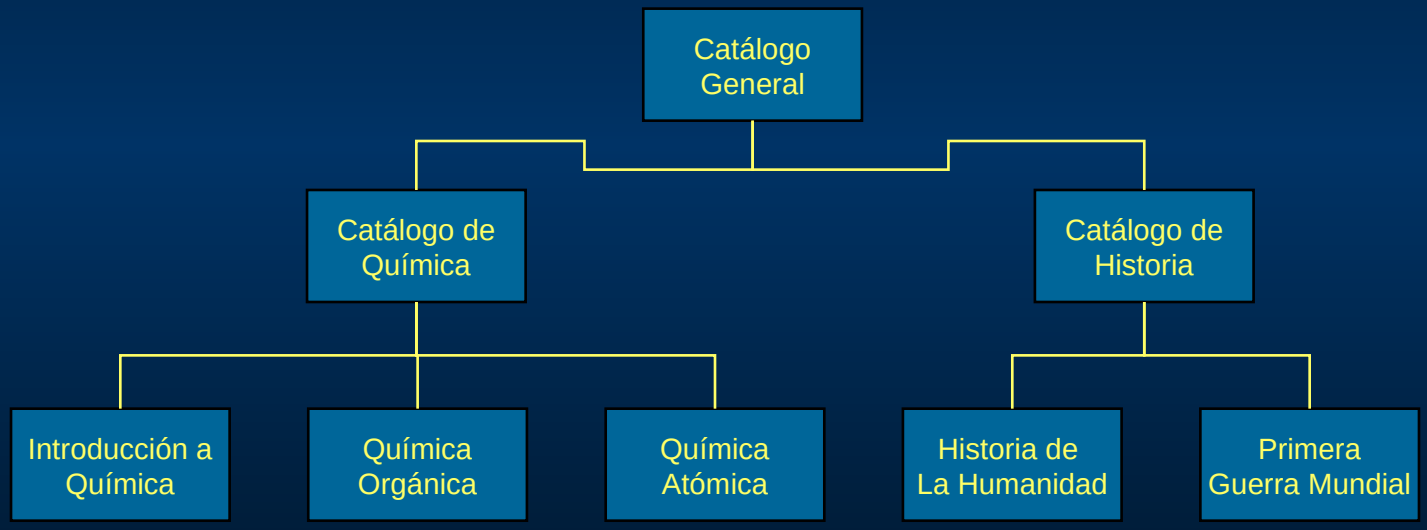
- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Bases de Datos.

    Las bases de datos se catalogan de acuerdo al tipo de relación que existe entre sus datos.

    - Jerárquicas
    - Network
    - Relacional

Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Bases de Datos.
    - **Jerárquicas**. Este tipo de base de datos, establece una relación de arriba - abajo entre los datos. Esta basada en una relación uno a uno o uno a muchos.

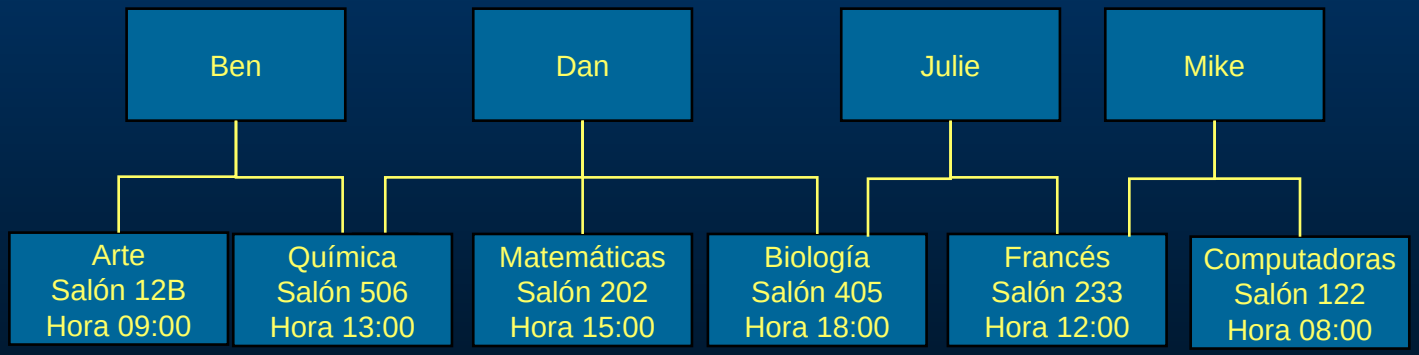


Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Bases de Datos.
    - Desventajas del tipo de Bases de Datos jerárquicas.
      - Estructura rígida.
      - Estructura Compleja.
      - Las relaciones son limitadas.

Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Bases de Datos.
    - **Red (Network)**. Este tipo de base de datos, no restringe la relación por lo que pueden establecerse relaciones desde uno a uno, uno a muchos y muchos a muchos.
      - El término de relación *owner-member*, es el mismo concepto de la relación padre-hijo.
      - El modelo de Red es más flexible que el jerárquico ya que permite la relación muchos a muchos, además de ser apropiada para el rápido procesamiento de altos volúmenes de información.



Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Bases de Datos.
    - Desventajas.
      - No permite la manipulación de datos por usuarios inexpertos.
      - Es una estructura estática, una vez definidas las relaciones no pueden ser cambiadas tan fácilmente, y son de alta complejidad.

Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Bases de Datos.
    - **Relacional**. Es un modelo de almacenamiento en tablas, que están compuestas de renglones y columnas.
      - La relación padre-hijo no existe en este tipo de Bases de datos, estas son mantenidas a través de la unión de columnas comunes **(CROSS REFERENCE**).

No-Emp.	Tarifa	Hrs. Trab.	Extra	Impuesto
SAID, FRANK	5.35	25	0	15%
DAVIS, JACKIE	8.25	42	2	25%
KOBREEK, NANCY	2.50	62	22	5%
BOVEN, MARY	4.50	12	0	17%

No-Emp.	Antigüedad	F.Nac.	Universidad
SAID, FRANK	14	7-12-1956	UNIVERSISDAD CALIFORNIA
DAVIS, JACKIE	1	21-03-1959	COLEGIO CARLLENGTON
KOBREEK, NANCY	5	13-12-1959	COLEGIO SMITH
BOVEN, MARY	15	4-08-1950	COLEGIO LAKE

CROSS REFERENCE

Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Bases de Datos.
    - Ventajas.
      - Fácil uso. Por el manejo de columnas y renglones.
      - Estructura de naturaleza dinámica. La cual puede ser modificada en cualquier momento para adicionar una columna, no causando gran impacto como lo sería en una estructura jerárquica o de red.
      - Fácil acceso a usuarios sin experiencia y desarrollo para sistemas sofisticados.
    - Desventajas.
      - Son menos estructuradas
      - El tiempo de procesamiento es más lento
      - No pueden manejar grandes cantidades de transacciones

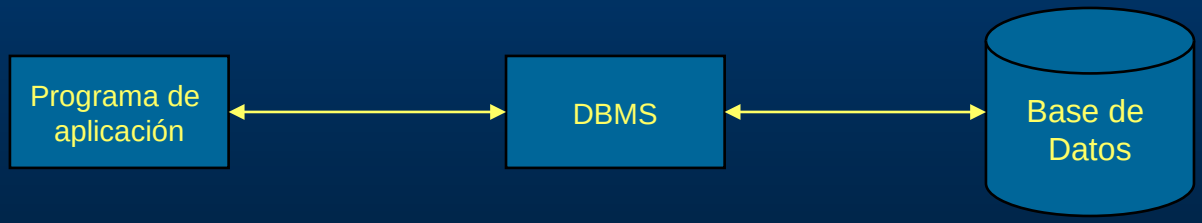
Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Tipos de Bases de Datos.

Tipo de Base de Datos	Estructura Predefinida	Vista Estática	Relación Entidad Padre-hijo	Rendimiento
Jerárquica	Si	Si	Uno a uno Uno a muchos	- Muy rápido - Manejo de alto volumen de transacciones
Network	Si	Si	Uno a uno Uno a muchos Muchos a muchos	- Rápido - Manejo de alto volumen de transacciones
Relacional	No	No	Relaciones no definidas	- Acceso medio - Manejo de volumen de transacciones medio.

Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Manejador de Base de Datos (DBMS).
    - Es un software que permite la comunicación entre los programas orientados al negocio y la base de datos.
    - Un DBMS permite concentrarse en las relaciones entre las diferente entidades de los datos, sin ser necesario conocer la organización de los datos.



Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Funciones de un DBMS.
    - *Almacenamiento de datos*. Controla el almacenamiento físico de las Bases de datos.
    - *Acceso de datos.* Controla cómo son accedidos los datos físicamente.
    - *Mantenimiento de los datos.* Esta incluye el mantenimiento de los datos como: actualización, inserción, borrado de datos.

Tópico 410 - Data Storage & Data Base Management System

- Sistemas Manejadores de Bases de Datos y Bases de Datos.
  - Ventajas de un DBMS.
    - *Velocidad*. Los datos pueden ser accedidos rápidamente.
    - *Seguridad*. El acceso a datos especiales es controlado
    - *Reduce la redundancia de datos*. Los datos sólo pueden ser almacenados una vez.
    - *Datos compartidos*. Más de una persona puede acceder a los mismos datos.
    - *Integridad de los datos*. Puede ser mantenida exacta y consistentemente.
    - *Almacenamiento lógico de los datos*. No es necesario conocer dónde se encuentran los datos almacenados, es suficiente hacer referencia lógica a los datos.

Módulo Específico

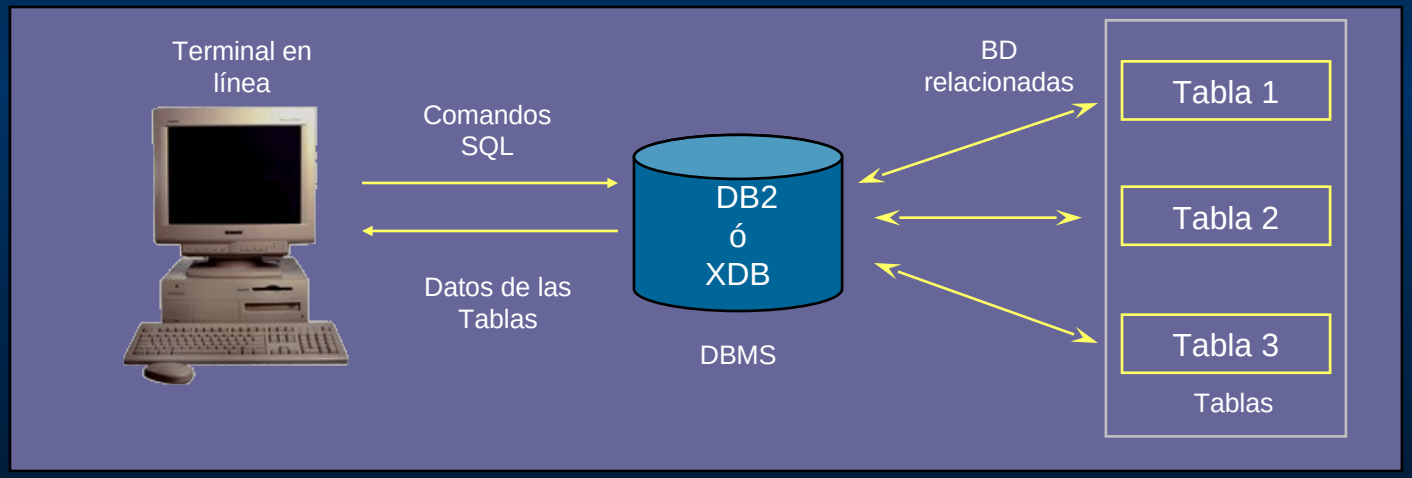
- Tópico 410 - Data Storage & Data Base Management System.
- **Tópico 420 - Accesing Data in a Data Base.**
- Tópico 430 - Accessing Data from within a Program.
- Estándares de programación SQL.

Tópico 420 - Accessing Data in a Data Base

- OBJETIVOS.
  - Escribir queries básicos de SQL usando el comando SELECT y sus variaciones.
  - Escribir queries usando la combinación básica con AND y OR
  - Mantenimiento y modificación de tablas usando comandos de INSERT, UPDATE, DELETE.

Tópico 420 - Accessing Data in a Data Base

- Introducción a SQL
  - SQL (Structure Query Lenguage). Es un lenguaje de manipulación de datos, el cual al ser utilizado en combinación con un manejador de base de datos permite acceder y manipular datos, así como definir datos (DB2).
  - Los datos contenidos en las tablas de la Base de datos son accedidos a través de QUERIES, que son sentencias casi en lenguaje inglés.

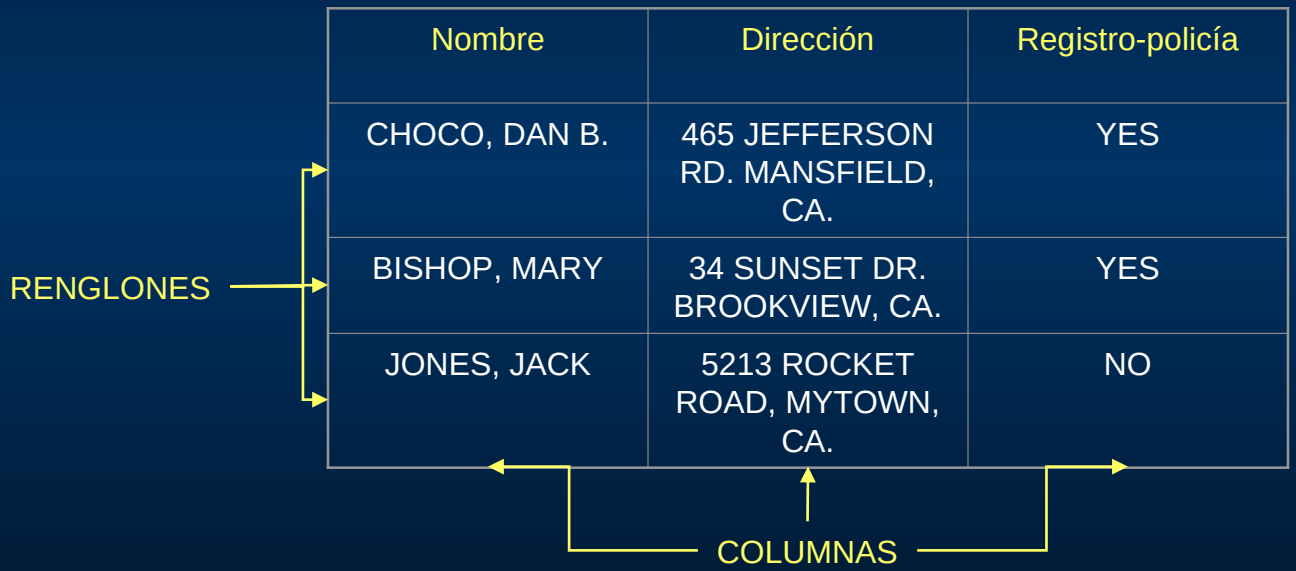


Tópico 420 - Accessing Data in a Data Base

- Introducción a DB2.
  - Las bases de datos relacionales comienzan a tener mayor auge en 1980, llegando a reemplazar a las bases de datos jerárquicas.
  - DB2 es una herramienta usada en las DBMS, corre bajo el sistema operativo MVS de IBM.
  - Características de DB2.
    - Utiliza lenguaje SQL (Structure Query Language), para acceder a los datos. En este lenguaje se distinguen dos sublenguajes:
      - ***Data Definition Language (DDL).*** Provee la definición o descripción de las tablas de las bases de datos. Manipula la estructura de las tablas.
      - ***Data Manipulation Language (DML)***. Realiza la manipulación o procesamiento de las tablas. Permite el acceso, actualización y mantenimiento de los datos.

Tópico 420 - Accessing Data in a Data Base

- Introducción a DB2.
  - Términos de DB2.
    - **Tabla**. Es similar a un archivo, se encuentra compuesta de columnas y renglones.



Tópico 420 - Accessing Data in a Data Base

- Introducción a DB2.
  - Términos de DB2.
    - **Vista**. Es una representación alternativa de los datos de una o más tablas.

Nombre	Dirección
CHOCO, DAN B.	465 JEFFERSON RD. MANSFIELD, CA.

VISTA

TABLA

Nombre	Dirección	Registro-policía
CHOCO, DAN B.	465 JEFFERSON RD. MANSFIELD, CA.	YES
BISHOP, MARY	34 SUNSET DR. BROOKVIEW, CA.	YES
JONES, JACK	5213 ROCKET ROAD, MYTOWN, CA.	NO

Tópico 420 - Accessing Data in a Data Base

- Introducción a DB2.
  - Términos de DB2.
    - **Query**. Es una sentencia SQL usada para acceder los datos de las tablas en una Base de datos.

Tópico 420 - Accessing Data in a Data Base

- Términos de Bases de datos

Términos de Bases de Datos Estándar	Términos de Bases de Datos Relacionales
1. Base de datos - Es una colección de archivos que contienen datos	1. Base de datos Relacional - es una colección de tablas
2. Archivo de datos - Es una colección de registros	2. Tabla - Es una colección de renglones y columnas.
3. Registro - Es un conjunto de campos.	3. Renglón - es un conjunto de valores de una columnas
4. Campos	4. Columnas

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL.
  - Son cuatro las funciones que permite realizar SQL.
    - Seleccionar (acceder) datos de las tablas (**SELECT**).
    - Insertar o adicionar datos a las tablas (**INSERT**).
    - Actualizar, modificar o dar mantenimiento a las tablas (**UPDATE**).
    - Eliminar o borrar datos de las tablas (**DELETE**).

Tópico 420 - Accessing Data in a Data Base

- Funciones Básicas de SQL

Tabla de Presidentes

PRE_NOMBRE	PRE_NAC	PRE_ANTI	PRE_EDAD	PRE_SEXO	PRE_OFIC
ENGEL L C	1900	8	79	M	TORONTO
LOPEZ R A	1927	2	46	F	MADRID
JOHNSON S L	1918	5	65	F	TORONTO
NEWTON D W	1923	5	-	M	CHICAGO
FORD S B	1923	2	-	M	NEW YORK
CARTIER J M	1934	4	-	F	GENOVA
RYAN J R	1921	8	-	M	LONDRES

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **SELECT**.
  - La sentencia **SELECT** permite acceder y seleccionar los datos de la tabla.

SELECT [*][*nombre_columna1,nombre_columnaN*] [*n*]
FROM tabla
WHERE *condición.*

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **SELECT**.
  - Selección de todas las columnas de una tabla.

QUERY	RESULTADO DEL QUERY

```
SELECT *
FROM   PRESIDENTES
```

PRE_NOMBRE	PRE_NAC	PRE_ANTI	PRE_EDAD	PRE_SEXO	PRE_OFIC
ENGEL L C	1900	8	79	M	TORONTO
LOPEZ R A	1927	2	46	F	MADRID
JOHNSON S L	1918	5	65	F	TORONTO
NEWTON D W	1923	5	-	M	CHICAGO
FORD S B	1923	2	-	M	NEW YORK
CARTIER J M	1934	4	-	F	GENOVA
RYAN J R	1921	8	-	M	LONDRES

- Evitar el uso de '*' a menos que se requiera, ya que éste implica tiempo de procesamiento.
- Los nombres de las columnas son conectados a través de '_'.

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **SELECT**.
  - Selecciona los datos de las columnas especificadas por su nombre.

QUERY	RESULTADO DEL QUERY

```
SELECT PRE_NOMBRE, PRE_SEXO
FROM   PRESIDENTES
```

PRE_NOMBRE	PRE_SEXO
ENGEL L C	M
LOPEZ R A	F
JOHNSON S L	F
NEWTON D W	M
FORD S B	M
CARTIER J M	F
RYAN J R	M

- Los nombres de las columnas deben ser separados por comas.
- El orden de las columnas aparece de acuerdo al especificado en el query.

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **SELECT**.
  - Selecciona los datos de las columnas especificadas por su posición.

QUERY	RESULTADO DEL QUERY

```
SELECT 5,1
FROM   PRESIDENTES
```

PRE_SEXO	PRE_NOMBRE
M	ENGEL L C
F	LOPEZ R A
F	JOHNSON S L
M	NEWTON D W
M	FORD S B
F	CARTIER J M
M	RYAN J R

    - El orden de las columnas aparece de acuerdo al especificado en el query.

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **SELECT DISTINCT**.
  - La sentencia **SELECT DISTINCT** permite seleccionar datos de la tabla y eliminar de la selección aquellos duplicados.

SELECT DISTINCT *nombre_columna*

FROM tabla

WHERE *condición.*

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **SELECT DISTINCT**.
  - Selección de una columna

QUERY	RESULTADO DEL QUERY
SELECT DISTINCT PRE_SEXO **FROM** PRESIDENTES	PRE_SEXO M F

  - Seleccionando dos columnas

QUERY	RESULTADO DEL QUERY
SELECT DISTINCT PRE_SEXO, PRE_OFIC **FROM** PRESIDENTES	PRE_SEXO / PRE_OFIC M / TORONTO F / MADRID F / TORONTO M / CHICAGO M / NEW YORK F / GENOVA M / LONDRES

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **WHERE**.
  - Permite establecer condiciones especificas de la selección de datos.

SELECT *nombre-columna*

FROM tabla

WHERE *nombre_columna1* [=, >,<,<>] *nombre_columna2, constante*

QUERY	RESULTADO DEL QUERY
SELECT PRE_NOMBRE **FROM** PRESIDENTES WHERE PRE_SEXO = 'M'	PRE_NOMBRE ENGEL L C NEWTON D W FORD S B RYAN J R

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **WHERE**.
  - Distinción de valores numéricos y alfanuméricos

Valor	Entre comillas	Búsqueda
Alfanuméricos	SI	Tienen que ser idéntico el carácter indicado al buscado
Numéricos	No	Los valores pueden ser equivalentes.

Valor	Representación Alfanumérica	Representación Numérica
123	'123'	123
ABC	'ABC'	No aplica
$123	'$123'	No aplica
1.2	'1.2'	1.2
1-2	'1-2'	No aplica

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **ORDER BY**.
  - La cláusula **ORDER BY** permite especificar el orden en el que deben ser listados los datos accedidos.

SELECT *nombre-columna*

FROM tabla

ORDER BY [*nombre_columna, N*]

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **ORDER BY**.
  - Ordenación por una columna

QUERY

```
SELECT   PRE_NOMBRE, PRE_SEXO,
         PRE_OFIC
FROM     PRESIDENTES
ORDER BY PRE_NOMBRE
```

RESULTADO DEL QUERY

PRE_NOMBRE	PRE_SEXO	PRE_OFIC
CARTIER J M	F	GENOVA
ENGEL L C	M	TORONTO
FORD S B	M	NEW YORK
JOHNSON S L	F	TORONTO
LOPEZ R A	F	MADRID
NEWTON D W	M	CHICAGO
RYAN J R	M	LONDRES

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **ORDER BY**.
  - Ordenación por más de una columna

QUERY	RESULTADO DEL QUERY

```
SELECT PRE_NOMBRE, PRE_SEXO,
       PRE_ANTI
FROM   PRESIDENTES
WHERE  PRE_NAC > 1921
ORDER BY PRE_SEXO, PRE_ANTI
```

PRE_NOMBRE	PRE_SEXO	PRE_ANTI
LOPEZ R A	F	2
CARTIER J M	F	4
FORD S B	M	2
NEWTON D W	M	5

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **ORDER BY**.
  - Los datos pueden ser ordenados ascendentemente (**ASC**)

QUERY	RESULTADO DEL QUERY
SELECT PRE_ANTI **FROM** PRESIDENTES **ORDER BY PRE_ANTI ASC**	PRE_ANTI 2 2 4 5 5 8 8

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **ORDER BY**.
  - Los datos pueden ser ordenados descendentemente (**DESC**)

QUERY	RESULTADO DEL QUERY
SELECT PRE_ANTI **FROM** PRESIDENTES **ORDER BY PRE_ANTI DESC**	PRE_ANTI 8 8 5 5 4 2 2

Tópico 420 - Accessing Data in a Data Base

- Funciones básicas de SQL - **ORDER BY**.
  - En vez de utilizar los nombres de las columnas con la cláusula **ORDER BY**, se pueden utilizar números que correspondan al orden en que las columnas se listan en la sentencia **SELECT**.

QUERY

```
SELECT PRE_NOMBRE, PRE_SEXO,
       PRE_ANTI
FROM   PRESIDENTES
WHERE PRE_NAC > 1921
ORDER BY  2, 3
```

RESULTADO DEL QUERY

PRE_NOMBRE	PRE_SEXO	PRE_ANTI
LOPEZ R A	F	2
CARTIER J M	F	4
FORD S B	M	2
NEWTON D W	M	5

Tópico 420 - Accessing Data in a Data Base

- Condiciones Múltiples.
  - Las condiciones usadas en la sentencia **WHERE** pueden ser combinadas con las cláusulas **AND** y **OR**.
    - **OR** - Se utiliza para vincular las condiciones de búsqueda en un query, la selección regresa los registros que cumplan con alguna de las condiciones especificadas.

QUERY	RESULTADO DEL QUERY

```
SELECT  PRE_NOMBRE, PRE_OFIC,
        PRE_SEXO
FROM    PRESIDENTES
WHERE   PRE_OFIC = 'TORONTO'
OR      PRE_OFIC = 'GENOVA'
```

PRE_NOMBRE	PRE_OFIC	PRE_SEXO
ENGEL L C	TORONTO	M
JOHNSON S L	TORONTO	F
CARTIER J M	GENOVA	F

Tópico 420 - Accessing Data in a Data Base

- Condiciones Múltiples.
  - **AND** - Se utiliza para vincular las condiciones de búsqueda en un query, la selección regresa los registros que cumplan con todas las condiciones especificadas.

Tabla de STAFF

STAFF_NOMBRE	STAFF_SALARIO	STAFF_PUESTO
SANDER	48357.50	DIRECTOR
HANES	30659.80	DIRECTOR
FRAYE	41150.00	DIRECTOR
SMITH	41225.00	VENTAS
QUILL	39818.00	DIRECTOR
GRAHAM	41000.00	VENTAS

QUERY	RESULTADO DEL QUERY

```
SELECT STAFF_NOMBRE,
       STAFF_SALARIO,
       STAFF_PUESTO
FROM   STAFF
WHERE  STAFF_PUESTO = 'DIRECTOR'
AND    STAFF_SALARIO > 40000
```

STAFF_NOMBRE	STAFF_SALARIO	STAFF_PUESTO
SANDER	48357.50	DIRECTOR
FRAYE	41150.00	DIRECTOR

Tópico 420 - Accessing Data in a Data Base

- Condiciones Múltiples.
  - ***OR y AND múltiples***. Se puede vincular más de una condición a través de OR y AND.

QUERY	RESULTADO DEL QUERY

```
SELECT PRE_NOMBRE, PRE_ANTI
FROM   PRESIDENTES
WHERE PRE_NOMBRE= 'JOHNSON S L'
OR       PRE_SEXO = 'F'
OR       PRE_ANTI > 4
```

PRE_NOMBRE	PRE_ANTI
ENGEL L C	8
LOPEZ R A	2
JOHNSON S L	5
NEWTON D W	5
CARTIER J M	4
RYAN J R	8

QUERY	RESULTADO DEL QUERY

```
SELECT PRE_NOMBRE, PRE_ANTI
FROM   PRESIDENTES
WHERE PRE_NOMBRE= 'JOHNSON S L'
AND      PRE_SEXO = 'F'
AND      PRE_ANTI > 4
```

PRE_NOMBRE	PRE_ANTI
JOHNSON S L	5

Tópico 420 - Accessing Data in a Data Base

- Condiciones Múltiples.
  - ***OR y AND combinados***. Se puede vincular más de una condición a través de OR y AND en forma combinada.
    - SQL evalúan primero las condiciones **AND,** en caso de que no sean utilizados paréntesis.

QUERY	RESULTADO DEL QUERY

```
SELECT FLIGHT, DEPARTS, DESTINATION
FROM   AIRLINE_SCHEDULE
WHERE  DEPARTS > 'PM1200'
AND    DESTINATION = 'NEW YORK'
OR     DESTINATION = 'NEWARK'
```

FLIGHT	DEPARTS	DESTINATION
101	PM1300	NEW YORK
116	PM1400	NEW YORK
142	AM0700	NEWARK
193	PM1400	NEWARK
184	PM1500	NEWARK

En el ejemplo las dos primeras expresiones son combinadas para crear la primera condición, la segunda condición sólo considera los vuelos con destino a NEWARK sin importar la hora, para tomar en cuenta la hora se requeriría el uso de paréntesis.

Tópico 420 - Accessing Data in a Data Base

- Paréntesis y cláusula **WHERE**.
  - Los paréntesis se pueden utilizar para eliminar el orden en que SQL evalúan las expresiones.
  - El uso de paréntesis con múltiples condiciones de búsqueda permite mejorar la legibilidad de la consulta.
  - El orden de búsqueda cuando se usan los paréntesis es:
    1. Se evalúan las expresiones encerradas en paréntesis
    2. Son consideradas las condiciones vinculadas con **AND**
    3. Se evalúan las condiciones vinculadas con **OR**

QUERY	RESULTADO DEL QUERY

```
SELECT FLIGHT, DEPARTS, DESTINATION
FROM   AIRLINE_SCHEDULE
WHERE  (DESTINATION = 'NEW YORK'
OR     DESTINATION = 'NEWARK')
AND    DEPARTS > 'PM1200'
```

FLIGHT	DEPARTS	DESTINATION
101	PM1300	NEW YORK
116	PM1400	NEW YORK
193	PM1400	NEWARK
184	PM1500	NEWARK

Tópico 420 - Accessing Data in a Data Base

- Uso de valores Nulos en condiciones.
  - Los valores nulos hacen referencia a un valor especial que indica ausencia de valor.
  - En algunos manejadores se representan por un guión '-'.
  - El que la columna de una tabla pueda o no contener valores nulos, dependerá de la definición que haya especificado el DBA cuando la tabla fue creada.

Por ejemplo:

Nulo

STAFF_NAME	STAFF_JOB	STAFF_SALARY	STAFF_COMM
HANES	CLERK	20659.00	-
ROTHMAN	SALES	16502.83	1152.00
NOONAN	CLERK	12508.20	206.60
NEWCOMBE	CLERK	11600.00	0.00

Tópico 420 - Accessing Data in a Data Base

- Uso de valores Nulos en condiciones.
  - Reglas de manejo para valores nulos:
    - Si una condición es aplicada en una columna que contiene valores nulos, los registros no son seleccionados.
    - Si se realiza un calculo utilizando una columna que contenga valores nulos, los registros que contienen nulos serán ignorados.

QUERY	RESULTADO DEL QUERY

```
SELECT STAFF_NAME,
       STAFF_JOB,
       STAFF_SALARY,
       STAFF_COM
FROM   STAFF
WHERE  STAFF_JOB = 'CLERK'
AND    STAFF_COM  < 1000.00
```

STAFF_NAME	STAFF_JOB	STAFF_SALARY	STAFF_COMM
NOONAN	CLERK	12508.20	206.60
NEWCOMBE	CLERK	11600.00	0.00

Tópico 420 - Accessing Data in a Data Base

- Uso de valores Nulos en condiciones.
  - Selección de valores nulos en una columna:

SINTAXIS	EJEMPLO
SELECT *columna1, columnaN* FROM *tabla* WHERE ***columna*** IS NULL	SELECT STAFF_NAME, STAFF_JOB, STAFF_SALARY, STAFF_COM FROM STAFF WHERE **STAFF_COM** IS NULL

  - Selección de valores no nulos en una columna:

SINTAXIS	EJEMPLO
SELECT *columna1, columnaN* FROM *tabla* WHERE ***columna*** IS NOT NULL	SELECT STAFF_NAME, STAFF_JOB, STAFF_SALARY, STAFF_COM FROM STAFF WHERE **STAFF_COM** IS NOT NULL

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **INSERT**
  - El comando **INSERT**, permite la adición nuevos registros a una tabla existente.

    INSERT INTO *tabla*

    VALUES (*valor1, valor2, valorN*)
  - Reglas para uso del comando **INSERT**:
    - Debe haber un valor para cada columna de la tabla.
    - Los valores deben estar listados en el orden en que las columnas aparecen en la tabla, en caso de no haber listado los campos.
    - Los valores deben estar separados por comas.
    - Cada campo alfanumérico (carácter) debe estar entre comillas.
    - Los valores numéricos no requieren comillas.

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **INSERT**
  - Inserción de registros sin especificación de columnas.

QUERY	RESULTADO DEL QUERY
INSERT INTO TELEFONOS VALUES ('ABRAMS', 'DIANE', 2037951234, 123)	APELLIDO | NOMBRE | NO_TELEF | EXT ABRAMS | DIANE | 2037951234 | 123

El orden de los valores debe ser el mismo en el que se encuentran organizadas las columnas de la tabla.

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **INSERT**
  - Inserción de registros especificando los nombres de columnas.

QUERY	RESULTADO DEL QUERY
INSERT INTO TELEFONOS (APELLIDO, NOMBRE, NO_TLEF, EXT) VALUES ('ABRAMS', 'DIANE', 2037951234, 123)	APELLIDO | NOMBRE | NO_TELEF | EXT ABRAMS | DIANE | 2037951234 | 123

De esta forma se pueden asignar los valores a los campos directamente en cualquier orden, sin llevar el mismo orden en que se encuentran organizadas las columnas de la tabla.

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **INSERT**
  - Inserción de registros con valores nulos.
    - El valor nulo sólo puede ser asignado para aquellas columnas hayan sido definidas para valores nulos.

QUERY	RESULTADO DEL QUERY
INSERT INTO TELEFONOS (APELLIDO, NOMBRE, NO_TLEF, EXT) VALUES ('ABRAMS', 'DIANE', 2037951234, **NULL**)	APELLIDO | NOMBRE | NO_TELEF | EXT ABRAMS | DIANE | 2037951234 | -

Cuando la columna maneja nulos, ésta puede ser omitida del query.

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **UPDATE**
  - El comando **UPDATE**, permite la adición nuevos registros a una tabla existente.

    UPDATE *tabla*

    SET *nombre_columna = valor*

    WHERE *condición.*
  - Si la cláusula **WHERE** es omitida, la actualización se realiza en todos los registros de la tabla.
  - Si se desea actualizar mas de una columna, se deberá separar por una coma y un espacio.

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **UPDATE**
  - Actualización de una columna.

QUERY	RESULTADO DEL QUERY
UPDATE TELEFONOS **SET** EXT = 111 **WHERE** APELLIDO = 'BAKER' AND NOMBRE = 'JOHN'	Antes de la actualización APELLIDO | NOMBRE | NO_TELEF | EXT BAKER | JOHN | 1234567890 | 999 DIETZ | ELLEN | 9876543210 | - Posterior a la actualización APELLIDO | NOMBRE | NO_TELEF | EXT BAKER | JOHN | 1234567890 | 111 DIETZ | ELLEN | 9876543210 | -

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **UPDATE**
  - Actualización de más de una columna.

QUERY	RESULTADO DEL QUERY
UPDATE TELEFONOS **SET** EXT = 111, NO_TELEF = 6666666666 **WHERE** APELLIDO = ‘ DIETZ’ AND NOMBRE = ‘ELLEN’	Antes de la actualización APELLIDO | NOMBRE | NO_TELEF | EXT BAKER | JOHN | 1234567890 | 999 DIETZ | ELLEN | 9876543210 | - Después de la actualización APELLIDO | NOMBRE | NO_TELEF | EXT BAKER | JOHN | 1234567890 | 999 DIETZ | ELLEN | 6666666666 | 111

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **UPDATE**
  - Actualización de TODOS los registros en una columna especifica.

QUERY	RESULTADO DEL QUERY
UPDATE TELEFONOS **SET** NO_TELEF = 4444444444	Antes de la actualización

APELLIDO	NOMBRE	NO_TELEF	EXT
BAKER	JOHN	1234567890	999
DIETZ	ELLEN	9876543210	-

Después de la actualización

APELLIDO	NOMBRE	NO_TELEF	EXT
BAKER	JOHN	4444444444	999
DIETZ	ELLEN	4444444444	-

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **DELETE**
  - El comando **DELETE** permite el borrado de registros de una tabla.

    DELETE FROM tabla

    WHERE condición.

  - Si la cláusula **WHERE** es omitida, todos los registros de la tabla son borrados. La tabla queda vacía.

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **DELETE**
  - Eliminación de registros con una condición especifica.

QUERY	RESULTADO DEL QUERY
DELETE FROM TELEFONOS **WHERE** NOMBRE = 'BAKER'	Antes de la actualización APELLIDO | NOMBRE | NO_TELEF | EXT BAKER | JOHN | 123456789 | 999 DIETZ | ELLEN | 9876543210 | - Después de la actualización APELLIDO | NOMBRE | NO_TELEF | EXT DIETZ | ELLEN | 9876543210 | -

Tópico 420 - Accessing Data in a Data Base

- Modificación de Tablas - **DELETE**
  - Eliminación de registros de una tabla

QUERY	RESULTADO DEL QUERY
DELETE FROM TELEFONOS	Antes de la actualización

APELLIDO	NOMBRE	NO_TELEF	EXT
BAKER	JOHN	123456789	999
DIETZ	ELLEN	9876543210	-

Después de la actualización

APELLIDO	NOMBRE	NO_TELEF	EXT

Módulo Específico

- Tópico 410 - Data Storage & Data Base Management System
- Tópico 420 - Accesing Data in a Data Base
- **Tópico 430 - Accessing Data from within a Program.**
- Estándares de programación SQL.

Tópico 430 - Accessing Data from within a Program

- OBJETIVOS.
  - Describir el funcionamiento de sentencias de SQL en un programa COBOL.
  - Describir como son definidas en la Working Storage las variables host y estructuras host, así como su utilización por estructuras SQL y programas COBOL.
  - Describir como interactúan el COBOL y SQL a través del uso de estructuras, variables host y manejo de errores.
  - Uso correcto de la sentencia SELECT para acceder un solo renglón de datos de una tabla SQL.
  - Explicar como interactúan los procesos de SQL y COBOL para acceder múltiples renglones de una tabla SQL.
  - Explicar el proceso de precompilación.

Tópico 430 - Accessing Data from within a Program

- Sentencias SQL .
  - Las sentencias SQL son precedidas por :
    - **EXEC SQL** que indica el inicio de la sentencia SQL
    - **END-EXEC** que indica el final de la sentencia SQL
    - La sintaxis de estas sentencias es:
      - Deben ser codificadas al inicio y al final de cada sentencia SQL
      - Ser codificadas entre las columnas 12 y 72 del programa COBOL
      - Terminar con un punto, a menos que esté contenida dentro de una estructura IF.

```
EXEC SQL
                SELECT  *
            INTO :DCLGEN-EMP
        FROM EMPLEADOS
END-EXEC.
```

Tópico 430 - Accessing Data from within a Program

- Variables Host.

  - Son los campos declarados en la Working Storage que permiten comunicarse a COBOL y SQL . Estos campos pueden ser usados directamente entre los dos lenguajes.

  - Las variables Host pueden ser usadas para:

    - Para realizar INSERT, UPDATE, DELETE de columnas.
    - Recibir valores de columnas.
    - Pueden ser usadas en cláusulas WHERE.

Tópico 430 - Accessing Data from within a Program

- Variables Host.

EMPLEADOS

EMP_ NOMBRE	EMP_ID	EMP_ SALARIO	EMP_ COM
JONES	12345678	90,000.00	9,000.00

```
WORKING-STORAGE SECTION.

01 CAMPOS-WORK.
      05 WS-HOST-VARIABLES.
            10 HOST-NOMBRE          PIC X(30).
            10 HOST-ID              PIC X(08).
            10 HOST-SALARIO         PIC S9(6)V9(2) COMP-3.
            10 HOST-COM             PIC S9(6)V9(2) COMP-3.
```

Tópico 430 - Accessing Data from within a Program

- Variables Host.

EMPLEADOS

EMP_ NOMBRE	EMP_ID	EMP_ SALARIO	EMP_ COMM
JONES	12345678	90,000.00	9,000.00

```
WORKING-STORAGE SECTION.

01 CAMPOS-WORK.
   05 WS-HOST-VARIABLES.
      10 HOST-NOMBRE  PIC X(30).
      10 HOST-ID      PIC X(08).
      10 HOST-SALARIO
          PIC S9(6)V9(2) COMP-3.
      10 HOST-COM
          PIC S9(6)V9(2) COMP-3.
```

```
EXEC SQL
     SELECT EMP_NOMBRE, EMP_ID, EMP_SALARIO,
            EMP_COMM
     INTO   :HOST-NOMBRE, :HOST-ID,
            :HOST-SALARIO, :HOST-COM
     FROM   EMPLEADOS
     WHERE  EMP_NOMBRE = :WS-NOMBRE
END-EXEC.
```

Tópico 430 - Accessing Data from within a Program

- Variables Host.
  - Reglas del uso de variables Host:
    - No necesariamente deben ser iguales a los nombres de las columnas.
    - Deben ser definidas en la Working Storage.
    - Cuando son usadas en una sentencia SQL deben ser precedidas por '**:**'.
    - La definición de las variables host debe coincidir con cada una de las columnas definidas en la tabla.
    - La estructura de las variables host sólo pueden contener como máximo un subnivel.

Tópico 430 - Accessing Data from within a Program

- Variables Host.
  - Compatibilidad del tipo de datos.

COBOL	SQL
WS-NUMBER PIC S9(6)V99 COMP-3.	DECIMAL (8,2)
WS-CARACTER PIC X(20)	CHAR (20)
WS-DATE PIC X(10)	DATE

La longitud y tipo definidos para las variables Host, debe ser idéntica a la definición de las columnas de la tabla DB2.

Tópico 430 - Accessing Data from within a Program

- Sentencia **INCLUDE**.
  - Para el caso de SQL, existe una sentencia que permite integrar copybooks a la Working Storage:

```
EXEC SQL
    INCLUDE copybook
END-EXEC.
```

  - La defínción de la tabla en DB2, así como su definición COBOL generalmente se encuentran en un solo copybook, los cuales son llamados DCLGEN.

Tópico 430 - Accessing Data from within a Program

- Contenido de un DCLGEN

```
EXEC SQL DECLARE AUTOS TABLE
    (
       AUTO_MATRICUL                  CHAR (10)
                                      NOT NULL,
       AUTO_MARCAUTO                  CHAR (15)
                                      NOT NULL,
       AUTO_MODEAUTO                  CHAR (10)
                                      NOT NULL,
       AUTO_PRECUNIT                  DECIMAL (10, 0),
       AUTO_FECHBAJA                  DATE
    )
    END-EXEC.


01  DCLAUTOS.
    03 AUTO-MATRICUL                  PIC X(10).
    03 AUTO-MARCAUTO                  PIC X(15).
    03 AUTO-MODEAUTO                  PIC X(10).
    03 AUTO-PRECUNIT                  PIC S9(10)V COMP-3.
    03 AUTO-FECHBAJA                  PIC X(10).
```

DEFINICION DB2

DEFINICION COBOL

Tópico 430 - Accessing Data from within a Program

- Comunicación y manejo de errores.
  - La ejecución de un query involucra un estatus de ejecución, a través de códigos de retorno o error.
  - Estos códigos de retorno son informados en la **SQLCA** (SQL Comunication Area), la cual es incluida en la Working Storage por medio de la sentencia **INCLUDE**.
  - El campo más usado de la SQLCA es:
    - **SQLCODE**. El cual indica si la sentencia SQL fue ejecutada o no correctamente. Los valores más frecuentes de este campo son:

Valor SQLCODE	Descripción
0	La sentencia SQL fue ejecuta exitosamente
+100	Indica que no se encontraron registros que cumplieran con las condiciones indicadas.
+##	Si el valor devuelto es distinto a los anteriores(<0 y <> +100) y es positivo indica que ha ocurrido un warning.
-###	Si el valor devuelto por la sentencia SQL es negativo(<0), indica que ha ocurrido un error .

Tópico 430 - Accessing Data from within a Program

- Comunicación y manejo de errores.

```
EXEC SQL
    SELECT EMP_ID, EMP_NOMBRE,
           EMP_SALARIO, EMP_COMM
    INTO  :EMP-NOMBRE, :EMP-EDAD,
          :EMP-SEXO
    FROM  EMPLEADO
    WHERE  DNI = :EMP-DNI
END-EXEC.

IF  SQLCODE  EQUAL ZERO          <-- Evalúa el código  0
    SET SI-ENCONTRADO  TO TRUE
ELSE
    IF  SQLCODE EQUAL WS-NOTFND  <-- Evalúa el código +100
        SET NO-ENCONTRADO TO TRUE
    ELSE
        MOVE WS-ERROR-SQL   TO LIN-INF-ERROR
        MOVE SQLCODE        TO LIN-CODE-SQL
        PERFORM 9999-ESCRIBE-MENSAJE-ERROR
           THRU 9999-ESCRIBE-MENSAJE-ERROR
    END-IF
END-IF.
```

Tópico 430 - Accessing Data from within a Program

- Comunicación y manejo de errores.
  - Sentencia **DISPLAY.**
    - Es una sentencia COBOL que permite desplegar mensajes de error durante la ejecución de un progrma.

Por ejemplo:

```
IF  SQLCODE  EQUAL ZERO
    SET SI-ENCONTRADO  TO TRUE
   ELSE
    IF  SQLCODE EQUAL WS-NOTFND
        DISPLAY 'REGISTRO NO ENCONTRADO: 'EMP-DNI
    ELSE
        MOVE WS-ERROR-SQL   TO LIN-INF-ERROR
        MOVE SQLCODE        TO LIN-CODE-SQL
        DISPLAY 'ERROR DE EJECUCION SQL: ' SQLCODE
    END-IF
END-IF.
```

Tópico 430 - Accessing Data from within a Program

- Selección de información simple.
  - A través de la sentencia **SELECT** se puede acceder a un sólo renglón de la tabla que cumpla con el criterio establecido.

```
EXEC SQL
     SELECT  nombre_columna
       INTO    :host-variable
       FROM    tabla
      WHERE  condición.
END-EXEC.
```

  - Condiciones de uso:
    - Asegurar que sólo un renglón de la tabla cumplirá con la condición de lo contrario se producirá un error de SQL.

Tópico 430 - Accessing Data from within a Program

- Selección de información múltiple.
  - Un **CURSOR,** es un método de SQL para acceso de uno o más renglones que cumplen un criterio establecido.
  - El uso de un cursor permite al programa colocar los datos en una tabla temporal llamada ACTIVE SET.
  - El uso de un cursor requiere de cuatro sentencias:
    - **DECLARE**.
    - **OPEN.**
    - **FETCH.**
    - **CLOSE.**

Tópico 430 - Accessing Data from within a Program

- Selección de información múltiple.
  - La sentencia **DECLARE**, indica a SQL las columnas que van a ser accedidas y las condiciones que se han de cumplir a través de la sentencia **SELECT**.

Sintaxis	Ejemplo

EXEC SQL
  DECLARE *nombre_cursor* **CURSOR FOR**
  SELECT *campo1, campo2, campoN*
  FROM *nombre_tabla*
  WHERE *condición*
END-EXEC.
	EXEC SQL
	  DECLARE EMPLEADO_CURSOR **CURSOR FOR**
	  SELECT EMP_ID, EMP_NOMBRE,
	  EMP_SALARIO, EMP_COMM
	  FROM EMPLEADOS
	  WHERE EMP_SALARIO >= :EMP-SALARIO
	END-EXEC.

Tópico 430 - Accessing Data from within a Program

- Selección de información múltiple.
  - La sentencia **OPEN**, activa el cursor declarado y se posiciona en los datos que cumplen con la condición (si existe) en la tabla temporal (Active Set).

Sintaxis	Ejemplo
EXEC SQL **OPEN** *nombre_cursor* **END-EXEC.**	**MOVE** WS-SALARIO **TO** EMP-SALARIO. **EXEC SQL** **OPEN** EMPLEADO_CURSOR **END-EXEC.**

  - Las sentencias **DECLARE** y **OPEN** siempre trabajan juntas.

Tópico 430 - Accessing Data from within a Program

- Selección de información múltiple.
  - La sentencia **FETCH**, accede uno a uno los registros que se encuentran en la tabla temporal (Active Set).

Sintaxis	Ejemplo

EXEC SQL
  FETCH *nombre_cursor*
  INTO *:HOST1, :HOST2, :HOSTn*
END-EXEC.
	EXEC SQL
	  FETCH EMPLEADO_CURSOR
	  INTO :EMP-ID, :EMP-NOMBRE,
	  :EMP:_SALARIO, :EMP_COMM
	END-EXEC.

Tópico 430 - Accessing Data from within a Program

- Selección de información múltiple.
  - La sentencia **CLOSE**, libera los recursos de la tabla de datos y la tabla temporal (Active Set).

Sintaxis	Ejemplo

EXEC SQL
  CLOSE *nombre_cursor*
END-EXEC.
	EXEC SQL
	  CLOSE EMPLEADO_CURSOR
	END-EXEC.

Tópico 430 - Accessing Data from within a Program

- Selección de información múltiple.

EMP_ID	EMP_ NOMBRE	EMP_ SALARIO	EMP_ COMM
12345678	JONES	90,000.00	9,000.00
12345890	TAYLOR	75,000.00	7,500.00
45544354	SMITH	82,500.00	8,250.00

DECLARE
OPEN

EMP_ID	EMP_ NOMBRE	EMP_ SALARIO	EMP_ COMM
12345678	JONES	90,000.00	9,000.00
12345890	TAYLOR	75,000.00	7,500.00
45544354	SMITH	82,500.00	8,250.00

FETCH

CLOSE

Tópico 430 - Accessing Data from within a Program

- Modificación de Tablas en COBOL con SQL.
  - Inserción de registros en una tabla.

```
EXEC SQL
    INSERT INTO nombre-tabla
    VALUES (:variable-host1,
            :variable-host2)
END-EXEC
```

Tópico 430 - Accessing Data from within a Program

- Modificación de Tablas en COBOL.
  - Actualización de registros en una tabla.

```
EXEC SQL
    UPDATE nombre-tabla
    SET  columna1 = :variable-host1,
         columna2 = :variable-host2
    WHERE condición
END-EXEC
```

Tópico 430 - Accessing Data from within a Program

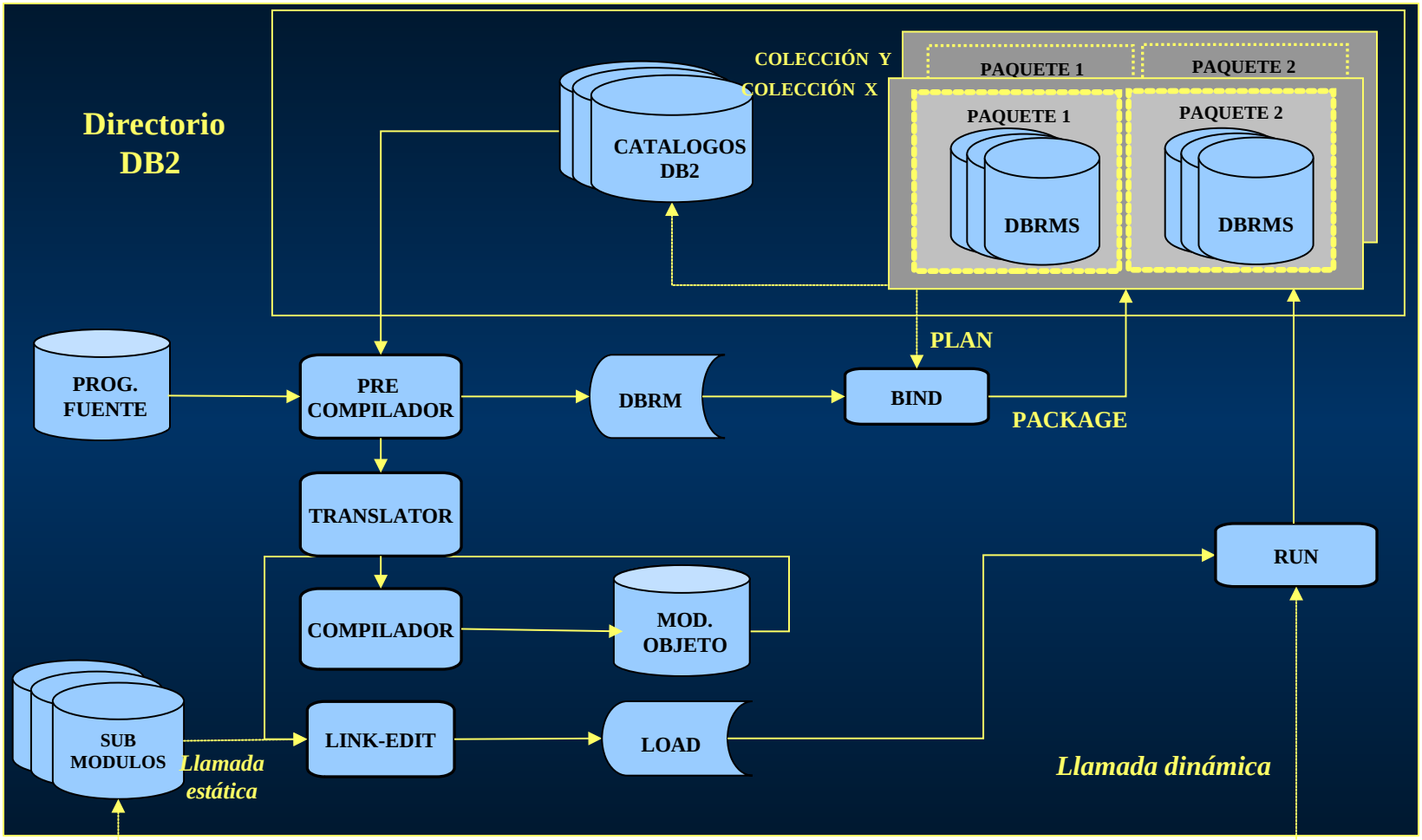
- Modificación de Tablas en COBOL.
  - Eliminación de registros en una tabla.

```
EXEC SQL
    DELETE FROM nombre-tabla
    WHERE condición
END-EXEC
```

Tópico 430 - Accessing Data from within a Program

- Proceso de Precompilación.
  - Durante el proceso de precompilación se realizan los siguientes pasos:
    - La precompilación se realiza antes del proceso de compilación.
    - Las sentencias SQL son traducidas por el precompilador, para después ser procesadas por COBOL, validando copybooks, sintaxis e inconsistencias.
    - Procesará todas las sentencias **INCLUDE**, a través de la librería SYSLIB.
    - Si el programa es precompilado sin errores, no significa que no existan estos, ya que el precompilador no se comunica con el DB2 para validar la existencia de las tablas o columnas que son mencionadas en el programa. El precompilador sólo verifica la sintaxis.

Tópico 430 - Accessing Data from within a Program



Módulo Específico

- Tópico 410 - Data Storage & Data Base Management System
- Tópico 420 - Accesing Data in a Data Base
- Tópico 430 - Accessing Data from within a Program.
- **Estándares de programación SQL.**

Estándares de Programación SQL

- Estándares de Programación SQL.
  - Utilización de Data Definition Leguaje (DDL).
    - Definición y creación de tablas, índices, etc., son responsabilidad del Administrador de Base de Datos.
  - En la manipulación de datos (DML) se deben seguir los siguientes estándares:
    - Codificar las sentencias SQL en el progragrama COBOL, entre las columnas 12 y 72.
    - Poner punto al final de la sentencia SQL (**EXEC SQL** y **END-EXEC**).
    - Si la sentencia SQL se encuentra dentro de un IF no se pondrá punto en la sentencia **END-EXEC**.
    - Las sentencias **INCLUDE** y **DECLARE** sean definidas en la Working Storage.

Estándares de Programación SQL

- Estándares de Programación SQL.
  - Utilizar una sangría de 4 caracteres a la derecha del **EXEC SQL**.
  - Alinear todos los verbos SQL y cláusulas.
  - Evitar el uso de predicados complejos '**OR**'
  - Especificar las columnas que sean requeridas únicamente.
- Uso de cursores.
  - Codificar un **OPEN** y un **CLOSE** por cada cursor.
  - Codificar las sentencias **DECLARE**, **OPEN**, **FETCH**, **CLOSE** en esta secuencia física dentro del programa.
- Manejo de errores.
  - Incluir la copy **SQLCA**.