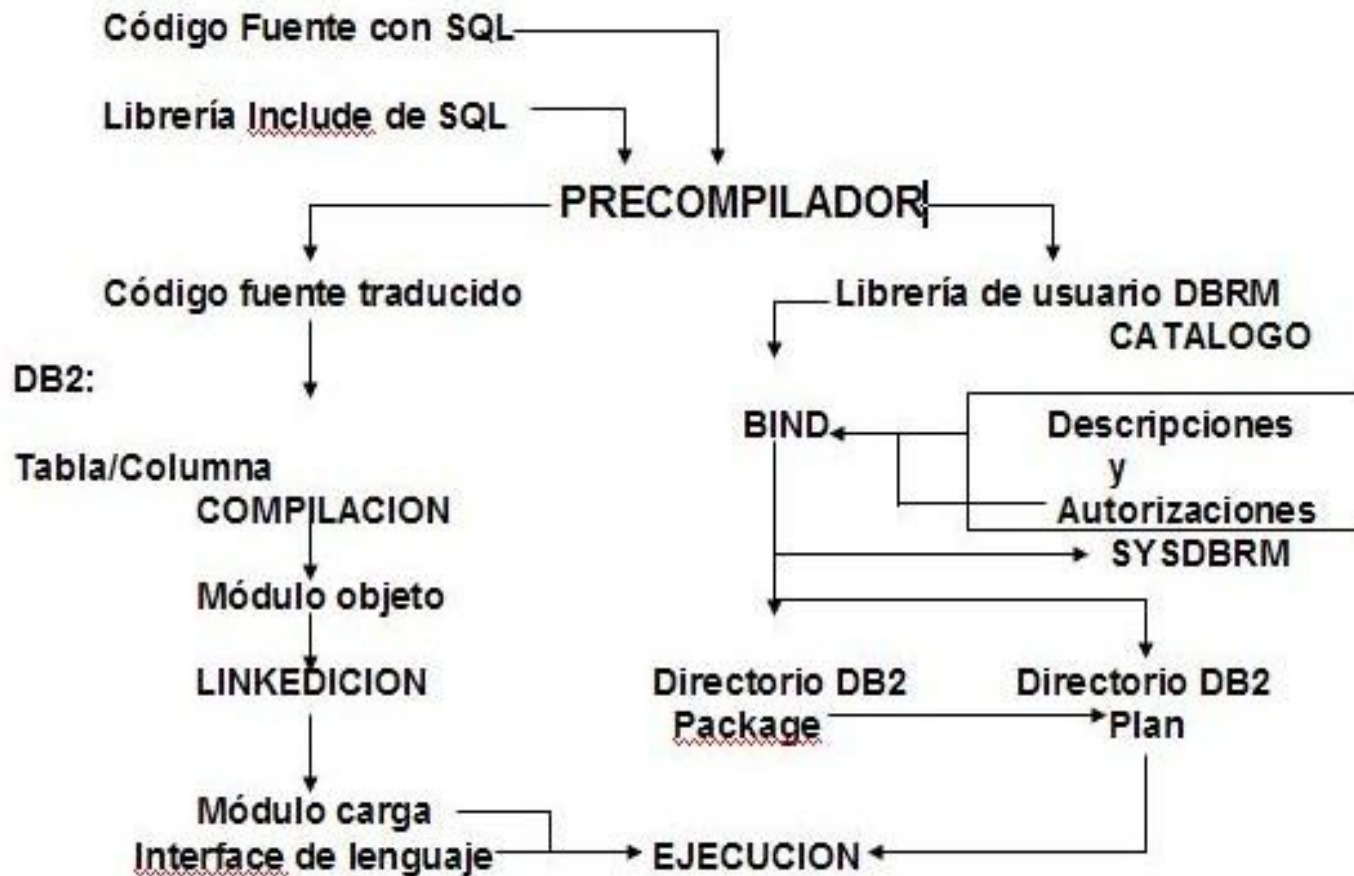


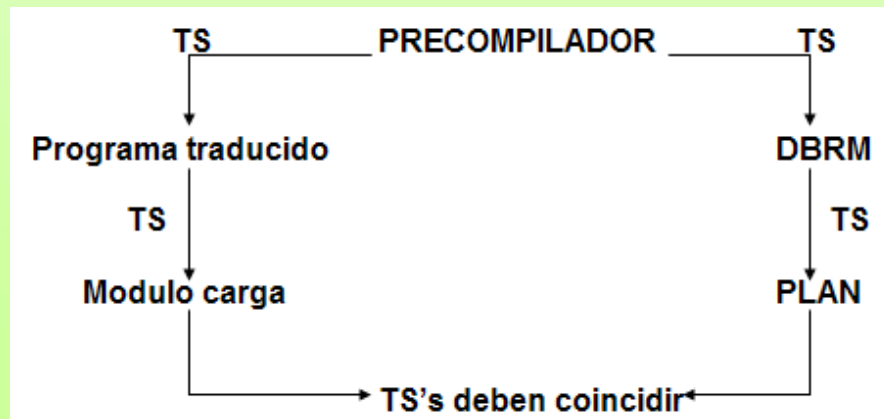
Desarrollo Cobol/DB2

Preparación de programas



Precompilador DB2

- Toma los postulados SQL del programa fuente
- Revisa la sintaxis de SQL
- Crea un programa fuente traducido de las sentencias SQL
- Crea el DBRM (Data Bases Request Module) y lo deja en la librería de usuario
- Los catálogos de DB2 no son accedados
- Timestamp en el precompilador DB2



Bind

- VALIDACION SOBRE:
 - Tablas/Columnas
 - Referencias
 - Descripciones
- AUTORIZACION
 - Accesos
 - Requerimientos de usuarios
- ESTRATEGIA DE ACCESO
 - Acceso físico
 - Características del requerimiento
- LOS DBRM pueden estar dentro de paquetes y/o Planes

SQL en comandos

- Instrucciones SQL codificadas en el programa con delimitadores
- SQL hace referencia a variables host
- Ejemplo:

```
EXEC SQL
  SELECT A, B INTO :X, :Y
  FROM T1
  WHERE C = :Z
END-EXEC.
```

- El programa debe contener el **SQL Communication Area**, que a su vez contiene el STATUS CODE
- Para procesar varios renglones, se utiliza un CURSOR
- En Cobol, SQL debe estar entre las columnas 12 y 72 del programa

SQL en comandos

INSTRUCCIÓN SQL:

```
UPDATE TEMPL  
SET NUMDEP = 'C02'  
WHERE NUMDEP = 'C01'
```

INSTRUCCIÓN SQL en un programa Cobol:

```
EXEC SQL  
    UPDATE TEMPL  
    SET NUMDEP = 'C02'  
    WHERE NUMDEP = 'C01'  
END-EXEC.
```

Variables Host

COBOL:

```
MOVE 'C01' TO W-DEP  
W-DEP
```

C01



SQL:

```
... WHERE NUMDEP = :W-DEP
```

- La variable host va precedida por el carácter dos puntos ":"
- Las variables host deben coincidir en el tipo de dato de la columna
- Estas no se podrán utilizar como nombres de objetos DB2

Variables Host

- Ejemplos:

SQL

```
INSERT INTO EMP  
      (NOEMP, APELLIDO)  
VALUES ('000190', 'JONES')
```

Cobol

```
EXEC SQL INSERT INTO EMP  
      (NOEMP, APELLIDO)  
VALUES (:W-NOEMP, :W-APELLIDO)  
END-EXEC.
```

W-NOEMP

000190

W-APELLIDO

JONES

Variables Host

- Ejemplos:

SQL

```
UPDATE EMP  
SET SALARIO = SALARIO * 1.05  
WHERE CODPUESTO = 54
```

Cobol

```
EXEC SQL  
    UPDATE EMP  
    SET SALARIO = SALARIO * :W-PORCENT  
    WHERE CODPUESTO = :W-CODIGO  
END-EXEC.
```

W-CODIGO 54

1.05
W-PORCENT

Definición de variables Host

DATOS NUMÉRICOS

SMALLINT	PIC S9(04) COMP.	2 bytes
INTEGER	PIC S9(09) COMP.	4 bytes
o INT		
DECIMAL(5,2)	PIC S9(03)V99 COMP-3	3 bytes
o DEC(5,2)		

DATOS ALFANUMÉRICOS

CHAR(10)	PIC X(10).	10 bytes
VARCHAR(80)	01 CAMPVAR.	
	05 CAMPOL	PIC S9(04) COMP.
	05 CAMPOC	PIC X(80).

DATE/TIME

DATE	PIC X(10).	10 bytes
TIME	PIC X(08).	8 bytes
TIMESTAMP	PIC X(26)	26 bytes

Indicador de nulidad

SELECT COLUMA INTO :W-A:W-AIND

W-A _____ W-AIND _____ -

- Un campo de indicador de nulidad (smallint) es requerido si la columna tiene característica allows NULL
- Si la columna contiene NULOS, el indicador de nulidad contendrá un valor negativo. El valor de la variable no se toca
- El indicador de nulidad puede ser puesto a valor negativo por programa para indicar NULL en UPDATE o INSERT

Indicador de nulidad

- Ejemplo:

```
EXEC SQL
```

```
    SELECT CODPUESTO, NUMDEP, TELEFONO
```

```
        INTO    :W-CPTO:W-CPTO-I    ,
```

```
                :W-DEP                ,
```

```
                :W-TEL:W-TEL-I
```

```
        FROM EMP WHERE NOEMP = :W-NOEMP
```

```
END-EXEC.
```

- CODPUESTO y TELEFONO son columnas con ALLOW NULL

Dando valor nulo a una columna

UPDATE:

```
EXEC SQL
        UPDATE EMP SET CODPUESTO = NULL
        WHERE NOEMP = :W-NOEMP
END-EXEC.
```

- Otra opción:

```
WORKING-STORAGE SECTION.
```

```
    01 W-CPTO-I      PIC S9(04) COMP.
    01 W-TEL-I       PIC S9(04) COMP.
```

```
.....
PROCEDURE DIVISION.
```

```
.....
    MOVE -1 TO W-CPTO-I.
    EXEC SQL
        UPDATE EMP
            SET CODPUESTO = :W-CPTO:W-CPTO-I
            WHERE NOEMP = :W-NOEMP
    END-EXEC.
```

Utilización de cursores

- Definiendo el cursor

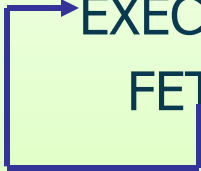
```
EXEC SQL  
  DECLARE CURSOR C1 CURSOR FOR  
  SELECT NOEMP, APELLIDO  
  FROM EMP  
  WHERE NUMDEP = :W-DEP  
END-EXEC.
```

- Abriendo el cursor

```
EXEC SQL OPEN C1 END-EXEC.
```

Utilización de cursores

- Obteniendo resultados uno a la vez



```
EXEC SQL  
    FETCH C1 INTO :W-NOEMP, W-APELLIDO  END-EXEC.
```

- Cerrando el cursor

```
EXEC SQL  CLOSE  C1  END-EXEC.
```

Utilización de cursores

- Borrando registros por medio de un cursor

```
EXEC SQL DECLARE C2 CURSOR ... ..
```

```
.....
```

```
EXEC SQL OPEN C2 END-EXEC.
```

```
.....
```

```
.....
```

```
EXEC SQL FETCH C2 INTO . . . . END-EXEC.
```

```
.....
```

```
EXEC SQL DELETE FROM EMP  
WHERE CURRENT OF C2 END-EXEC.
```

```
.....
```

```
EXEC SQL CLOSE C2 END-EXEC.
```


Actualizando registros vía un cursor

```
EXEC SQL DECLARE C3 CURSOR FOR  
    SELECT NOEMP, APELLIDO FROM EMP  
    WHERE NUMDEP = :W-DEP FOR UPDATE OF APELLIDO  
END-EXEC.
```

```
.....  
EXEC SQL OPEN C3 END-EXEC.
```

```
.....
```

```
EXEC SQL FETCH C3 INTO :W-NOEMP, :W-APELLIDO END-EXEC.
```

```
.....  
EXEC SQL UPDATE EMP SET APELLIDO = :W-APE-NUEVO  
    WHERE CURRENT OF C3 END-EXEC.
```

```
.....  
EXEC SQL CLOSE C3 END-EXEC.
```

Actualizando registros vía un cursor

- Si la instrucción SELECT de un cursor contiene ORDER BY, GROUP BY, DISTINCT, UNION, funciones o JOIN, no se pueden utilizar los postulados FOR UPDATE OF y DELETE WHERE CURRENT OF
- COMMIT puede cerrar cursores abiertos
- CLOSE de cursor no hace COMMIT

Commit/Rollback

- COMMIT
 - Indica la terminación de una unidad de trabajo
 - Los cambios a datos se hacen permanentes
 - Todos los LOCK de páginas son liberados
 - Los bloqueos a TABLESPACE son liberados si existe el postulado RELEASE(COMMIT) en el BIND
- ROLLBACK
 - La unidad de trabajo actual es abandonada
 - Los cambios a los datos son hechos hasta el último COMMIT
 - Todos los LOCK de página son liberados
 - Los LOCK de TABLESPACE son liberados si existe el postulado RELEASE(COMMIT) en el BIND
 - Los cursores son cerrados

SQLCA (SQL Communication area)

- Provee información que describe el estatus de la última instrucción SQL ejecutada
- Debe ser examinada por el programa después de cada requerimiento
- Es requerida y debe ser nombrada SQLCA
- Se puede incorporar al programa con la instrucción de pre-compilación:

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

FORMATO DEL SQLCA:

SQLCAID	CHAR (08)
SQLCABC	INTEGER
SQLCODE	INTEGER
SQLERRM	VARCHAR (70)
SQLERRP	CHAR (08)
SQLERRD (1)	INTEGER
SQLERRD (2)	INTEGER
SQLERRD (3)	INTEGER
SQLERRD (4)	INTEGER
SQLERRD (5)	INTEGER
SQLERRD (6)	INTEGER
SQLWARNO, SQLWARN1, SQLWARN2, SQLWARN3, SQLWARN4, SQLWARN5, SQLWARN6, SQLWARN7, SQLWARN8, SQLWARN9,	
SQLWARNA	(Todos los SQLWARNx son CHAR(01))
SQLSTATE	CHAR (05)

Códigos de SQLCA

CONDICION	SQLCODE	SQLWARNO	STATUS REQUERIMIENTO
ERROR	< 0		FALLA
WARNING	> 0 PERO NO + 100	"W"	SATISFECHO PERO CON UNA CONDICION ESPECIAL
NOT FOUND	+ 100		SIN DATOS O INEXISTENTE
CORRECTO	0	" "	CORRECTO

- El SQLERRD(3) indica el número de líneas cambiadas por INSERT, UPDATE o DELETE
- SQLWARNO Contiene:
 - ' ' = Si los SQLWARN1-A tienen blancos
 - 'W' = Si hay uno o más SQLWARN1-A

Códigos de SQLCA:

- SQLWARN1. String truncada en la variable host
- SQLWARN2. Valor(es) nulos eliminados en evaluación o función de columna
- SQLWARN3. Número de columnas mayor al número de variables host
- SQLWARN4. UPDATE o DELETE sin cláusula WHERE
- SQLWARN5. Instrucción SQL no válida en DB2
- SQLWARN6. Valor de DATE o TIMESTAMP ajustado para corregir un resultado válido de fecha

Códigos de SQLCA:

- SQLWARN7. Uno o más dígitos no-cero fueron eliminados de una fracción de un número usado en operación de multiplicación o división
- SQLWARN8. Un carácter que no pudo ser convertido, fue reemplazado con un carácter sustituto
- SQLWARN9. Excepciones aritméticas fueron ignoradas durante COUNT DISTINCT
- SQLWARNA. No aplica
- SQLSTATE. Contiene el código de retorno de la más reciente ejecución de un SQL

FIN DE MODULO

Derechos de autor

- Este producto has sido elaborado por
- Jorge Godínez Rodríguez.
- Derechos reservados
 - Prohibida su reproducción parcial o total