SQL INMERSO

- *SQL* inmerso, permite utilizar el lenguaje *SQL* para realizar accesos a una base de datos desde un programa escrito en un lenguaje de alto nivel (C, Pascal, COBOL, ...).
- Nosotros vamos a utilizar *SQL* inmerso en *C*
- Escribiremos un programa "casi *C*" con extensión *.pc*, que incluirá tanto las sentencias habituales de *C*, como sentencias *SQL* inmersas
- Este programa deberá ser preprocesado por la utilidad ***Pro*C/C++***, que, genera un programa con extensión *.c*

SQL INMERSO. EDITAR, PREPROCESAR, COMPILAR Y ENLAZAR

Los pasos que hay que dar para generar un ejecutable son:

1. Escribir el programa con el editor *vi* y ponerle extensión *.pc*:
   vi programa.pc

2. Preprocesarlo con *Pro*C/C++*:
   proc MODE=ANSI iname=programa.pc

3. Compilar y enlazar

SQL INMERSO. PREPROCESAR, COMPILAR Y ENLAZAR

FICHERO MAKEFILE

Se usará la utilidad *make* con fichero *makefile* que se facilita para crear el ejecutable a partir del fuente *.pc*

También se puede escribir un *script* llamado *compila* con el siguiente contenido:

```
gmake –f $HOME/bin/makefile build EXE=$1 OBJS=$1.o
```

Así, para preprocesar, compilar y enlazar:

compila programa

Se recomienda organizar estos archivos de la siguiente manera:

- Crear un directorio *bin* en la cuenta del usuario en olivo
- Copiar los archivos *makefile* y *compila* en *bin*
- Añadir el directorio *bin* a la variable de entorno *PATH*

SQL INMERSO. SENTENCIAS

EXEC SQL <sentencia SQL>;

- Los programas .*pc* incluyen una o varias sentencias de *SQL* inmerso
- Las sentencias de *SQL* inmerso pueden realizar accesos a una base de datos, o pueden ser simplemente declarativas
- Todas las sentencias que impliquen acceso a una base de datos deben ejecutarse dentro de una conexión

SQL INMERSO. CONEXIÓN Y DESCONEXIÓN DE UNA BASE DE DATOS

```
EXEC SQL CONNECT :username IDENTIFIED BY :password;
```

```
char oracleid [] = "/";
EXEC SQL CONNECT :oracleid;
```

```
EXEC SQL COMMIT WORK RELEASE;
```

```
EXEC SQL ROLLBACK WORK RELEASE;
```

SQL INMERSO. VARIABLES DEL LENGUAJE

- Algunas de las sentencias *SQL* hacen referencia a variables del lenguaje (variables declaradas como variables *C*)
- Se deben cumplir las siguientes condiciones:
  1. Las variables deben estar definidas en una sección de declaración
  2. En las sentencias *SQL*, dichas variables aparecerán precedidas de dos puntos ":"
- Una sección de declaración es un conjunto de sentencias de definición de variables y tipos comprendidas entre las sentencias
  EXEC SQL BEGIN DECLARE SECTION; y
  EXEC SQL END DECLARE SECTION;
- Las secciones de declaración pueden ser locales a una función *C* o globales del programa

SQL INMERSO. VARIABLES DEL LENGUAJE. EJEMPLO

```
int main ()
{
int c_editorial;
EXEC SQL BEGIN DECLARE SECTION;
   int a_edicion;
   char tit[61], oracleid[]="/";
EXEC SQL END DECLARE SECTION;
EXEC SQL CONNECT :oracleid;
EXEC SQL SELECT titulo, ano_edicion
   INTO :tit, :a_edicion
   FROM libro
   WHERE ISBN= '123456';
.......
EXEC SQL COMMIT WORK RELEASE;
}
```

SQL INMERSO. VARIABLES DEL LENGUAJE. EJEMPLO

```
int main ()
{
int c_editorial;
EXEC SQL BEGIN DECLARE SECTION;
   int a_edicion;
   char tit[61], oracleid[]="/";
EXEC SQL END DECLARE SECTION;
EXEC SQL CONNECT :oracleid;
EXEC SQL SELECT titulo, ano_edicion
   INTO :tit, :a_edicion
   FROM libro
   WHERE ISBN= '123456';
.......
EXEC SQL COMMIT WORK RELE
}
```

Sección de Declaración

Sentencia *SQL*

SQL INMERSO. VARIABLES DEL LENGUAJE. EJEMPLO

```
int main ()
{
int c_editorial;
EXEC SQL BEGIN DECLARE SECTION;
   int a_edicion;
   char tit[61], oracleid[]="/";
EXEC SQL END DECLARE SECTION;
EXEC SQL CONNECT :oracleid;
EXEC SQL SELECT titulo, ano_edicion
   INTO :tit, :a_edicion
   FROM libro
   WHERE ISBN= '123456';
.......
EXEC SQL COMMIT WORK RELEASE;
}
```

Variables de C

SQL INMERSO. VARIABLES DEL LENGUAJE. EJEMPLO 2

```
int main ()
{
int c_editorial;
EXEC SQL BEGIN DECLARE SECTION;
   int a_edicion;
   char tit [61], oracleid[]="/";
EXEC SQL END DECLARE SECTION;
EXEC SQL CONNECT :oracleid;
EXEC SQL SELECT titulo, ano_edicion,
cod_editorial
   INTO :tit, a_edicion, c_editorial
   FROM libro WHERE ISBN= '123456';
. . . . . .
EXEC SQL COMMIT WORK RELEASE;
}
```

SQL INMERSO. VARIABLES DEL LENGUAJE. EJEMPLO 2

```
int main ()
{
int c_editorial;
EXEC SQL BEGIN DECLARE SECTION;
   int a_edicion;
   char tit [61], oraclaid[]="/";
EXEC SQL END DECLARE
EXEC SQL CONNECT :ora
EXEC SQL SELECT titul
cod_editorial
   INTO :tit, a_edicion, c_editorial
   FROM libro WHERE ISBN= '123456';
. . . . . .
EXEC SQL COMMIT WORK RELEASE;
}
```

¡¡¡ ERROR 1 !!!

Faltan los dos puntos en *a_edicion*

SQL INMERSO. VARIABLES DEL LENGUAJE. EJEMPLO 2

```
int main ()
{
int c_editorial;
EXEC SQL BEGIN DECLARE SECTION;
   int a_edicion;
   char tit [61];
EXEC SQL END DE
EXEC SQL CONNEC
EXEC SQL SELECT
cod_editorial
   INTO :tit, a_edicion, c_editorial
   FROM libro WHERE ISBN= '123456';
. . . . . .
EXEC SQL COMMIT WORK RELEASE;
}
```

¡¡¡ ERROR 2 !!!

Faltan los dos puntos en *c_editorial*

c_editorial no está en la Sección de Declaración

SQL INMERSO. VARIABLES DEL LENGUAJE Y FUNCIONES

- Si una función recibe como argumento un dato que ha de utilizarse en una sentencia *SQL*, previamente deberá copiar el argumento en una variable local definida en una sección de declaración

```
int cuentalibros (int cod_sucursal){
EXEC SQL BEGIN DECLARE SECTION;
     int n;
     int cod;
EXEC SQL END DECLARE SECTION;
cod=cod_sucursal;
exec sql select count(*) into :n
from dispone where cod_suc=:cod;
return n;
}
```

SQL INMERSO. VARIABLES DEL LENGUAJE Y FUNCIONES

- Si una función recibe como argumento un dato que ha de utilizarse en una sentencia *SQL*, previamente deberá copiar el argumento en una variable local definida en una sección de declaración

```
int cuentalibros (int cod_sucursal){
EXEC SQL BEGIN DECLARE SECTION;
     int n;
     int cod;
EXEC SQL END DECLARE SECTION;
cod=cod_sucursal;
exec sql select count(*) into :n
from dispone where cod_suc=:cod;
return n;
}
```

SQL INMERSO. VARIABLES DEL LENGUAJE Y FUNCIONES

- Compatibilidad de tipos SQL – C

Tipo SQL	Tipo C, Pro*C/C++
CHAR (X)	char[n] VARCHAR[n] int short long float double
NUMBER	int
NUMBER (p,s)	short int long float double char char[n] VARCHAR[n]
DATE	char[n] VARCHAR[n]
LONG	char[n] VARCHAR[n]

SQL INMERSO. SECCIÓN DE COMUNICACIÓN *SQLCA*

```
EXEC SQL INCLUDE SQLCA;
```

- Se debe escribir al principio del programa
- Crea una variable de tipo *struct* llamada *sqlca* que contiene información sobre el último acceso a la base de datos
- Algunos de sus miembros son:
  - *sqlca.sqlcode*: código de error
  - *sqlca.sqlerrm.sqlerrmc*: mensaje de error
  - *sqlca.sqlerrd[2]*: número de filas afectadas por la sentencia
- Algunos valores de *sqlca.sqlcode* son:
  - 0 No ha habido error. Operación correcta
  - 100 No hay datos

```
printf ("\n%d %s\n", sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc);
```

SQL INMERSO. ACCESO A LA BASE DE DATOS SIN CURSORES.

SENTENCIA *SELECT*

EXEC SQL
SELECT *expresión* {, *expresión*}
INTO :*variable* {, :*variable*}
FROM *tabla* {, *tabla*}
[WHERE *cláusula*]
[GROUP BY *cláusula*]
[HAVING *cláusula*]
[ORDER BY *cláusula*]

- Sólo permite recuperar **una fila** de la base de datos
- Las columnas recuperadas se almacenan en las variables de la cláusula *INTO*
- Debe haber una correspondencia de número, orden y tipo entre las columnas de la cláusula *SELECT* y las variables de *INTO*

SQL INMERSO. ACCESO A LA BASE DE DATOS SIN CURSORES.

SENTENCIA *SELECT*

```
exec sql select nombre into :nom from lector
where codigo = :codigo ;

exec sql select count(*) into :n  from libro
where ano_edicion > 2010;

exec sql select * into :autor from autor
where codigo = :codigo;

exec sql select direccion into :lector.DIRECCION
from lector where codigo = :codigo;
```

SQL INMERSO. ACCESO A LA BASE DE DATOS SIN CURSORES.

SENTENCIA *SELECT.* EJEMPLO

```
#include <stdio.h>
exec sql include sqlca;
int main () {
exec sql begin declare section;
        struct lector_
        { int CODIGO; char NOMBRE[21], APE_1[21], /* resto de miembros de lector_*/ } lector;
        int cod; char oracleid[]="/";
exec sql end declare section;

printf ("Introduzca el codigo del lector a consultar: ");
scanf ("%d", &cod);

exec sql connect :oracleid;
printf ("CONEXION: %d, %s\n", sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc);

exec sql select * into :lector  from lector where codigo = :cod;
printf ("SELECT: %d, %s\n", sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc);
if (sqlca.sqlcode == 0)
        {printf ("APELLIDOS Y NOMBRE: %s %s, %s\n",
                lector.APE_1, lector.APE_2, lector.NOMBRE);
        printf ("DIRECCION: %s. %s. %s\n", lector.DIRECCION, lector.POBLACION,
                lector.PROVINCIA);
        printf ("FECHA NACIMIENTO: %s\n", lector.FECHA_NAC);
        }
else printf ("No existe ningun lector con CODIGO %d\n", cod);

exec sql commit work release;
}
```

SQL INMERSO. ACCESO A LA BASE DE DATOS SIN CURSORES. SENTENCIAS *UPDATE*, *INSERT* y *DELETE*

```
EXEC SQL
    UPDATE tabla
    SET columna = expresión {, columna = expresión }
    [ WHERE condición ] ;
```

```
EXEC SQL
    INSERT INTO tabla [(lista_columnas)]
    VALUES (lista_valores);

EXEC SQL
    INSERT INTO tabla [(lista_columnas)] subselect;
```

```
EXEC SQL
    DELETE  FROM tabla [ WHERE condición ] ;
```

SQL INMERSO. ACCESO A LA BASE DE DATOS SIN CURSORES. SENTENCIAS *UPDATE*, *INSERT* y *DELETE*

```
EXEC SQL
    UPDATE tabla
    SET columna = expresión {, columna = expresión }
    [ WHERE condición ] ;
```

```
EXEC SQL
    INSERT INTO tabla [(lista_columnas)]
    VALUES (lista_valores);

EXEC SQL
    INSERT INTO tabla [(lista_columnas)] subselect;
```

Pueden contener variables del lenguaje

```
EXEC SQL
    DELETE  FROM tabla [ WHERE condición ] ;
```

SQL INMERSO. ACCESO A LA BASE DE DATOS SIN CURSORES. SENTENCIAS *UPDATE*, *INSERT* y *DELETE*

```
exec sql update lector set direccion = 'BENAVENTE 7'
where codigo = :codigo ;

exec sql insert into escribe
values (:varautor.CODIGO, :varlibro.ISBN);

exec sql delete from sucursal where provincia = :nombreprov;
```