# Pequeña referencia sobre progamación con MySQL

## Funciones de control de flujo

Son las que van dentro del *SELECT*. Están vistas en el tema 1.

```
IF(expr1,expr2,expr3)
SELECT Nombre AS 'País',
       IF(EsperanzaVida IS NULL,
          'El país no se ha independizado',
           IF(ROUND(EsperanzaVida) % 2 = 0, 'Es par', 'Es impar'))
       AS 'Tipo de esperanza de vida'
FROM Pais
LIMIT 10;
```

Si expr1 es TRUE (expr1 <> 0) entonces IF() retorna expr2; de otro modo (expr1 = 0 OR expr1 IS NULL) retorna expr3.
La orden IF trata los valores nulos como si fuesen falsos y deberemos llevar mucho cuidado con este comportamiento.

```
CASE value
    WHEN [compare-value] THEN result
   [WHEN [compare-value] THEN result ...]
   [ELSE result]
END

SELECT Lengua, CASE EsOficial
                    WHEN 'T' THEN 'Es oficial'
                    WHEN 'F' THEN 'No es oficial'
                    WHEN 'X' THEN 'Otro tipo'  -- X no existe en nuestra BD
                    ELSE 'N/A'                 -- Los nulos si existiesen
               END AS 'Estado de la lengua'
FROM LenguaPais
WHERE CodigoPais = 'ESP';
```

```
CASE
    WHEN [condition] THEN result
   [WHEN [condition] THEN result ...]
   [ELSE result]
END

SELECT Pais.Nombre AS 'País',
       CASE
          WHEN AnyIndep >= 0    THEN CONCAT(Anyindep,' d.C.')
          WHEN AnyIndep <  0    THEN CONCAT(ABS(Anyindep),' a.C.')
          WHEN AnyIndep IS NULL THEN 'n/a'
       END AS 'Año de independencia'
FROM   Pais
WHERE  Continente='Africa'
ORDER BY AnyIndep;
```

Si no hay ninguna opción válida, *CASE* devuelve NULL:

```
SELECT CASE WHEN 3<2 THEN 'UNO' WHEN 4<3 THEN 'DOS' END -- Da NULL
```

# Tipos de datos

```
INT, VARCHAR(255), BOOLEAN, CHAR, FLOAT, DOUBLE, DATE, DATETIME, TIME, YEAR, TEXT, ENUM('T','F')
```

# Constantes

```
TRUE, FALSE ;
```

# Variables de usuario

- Empiezan por @
- No es necesario declararlas. Su tipo es el mismo que el del valor que contienen.
- son específicas de la conexión. Todas las variables de un cliente son automáticamente liberadas cuando ese cliente abandona la conexión

  ```
  SET @nombre_var = expresión
  ```
- Una variable de usuario también puede recibir valores en otras sentencias que no sean SET. En este caso, el operador de asignación debe ser := y no = porque = se considera operador de comparación en otras sentencias que no sean SET:

```
SELECT Continente,
       @t1 := ROUND(AVG(Poblacion)),
       @t2 := AnyIndep, ROUND(@t1 / @t2)
FROM Pais
GROUP BY Continente;
```

No se puede asignar un valor a una variable y leer su valor dentro del mismo SELECT:

```
SELECT Continente, @t1:=ROUND(AVG(Poblacion)), @t1 + COUNT(*)
FROM Pais
GROUP BY Continente;
-- Da valores incorrectos porque SELECT se evalua de derecha a izquierda


SET @PoblacionMundial= (SELECT SUM(Poblacion) FROM Pais);
SELECT Nombre,
       CONCAT(ROUND(Poblacion / @PoblacionMundial * 100), ' %')
       AS 'Porcentaje de la población mundial'
FROM   Pais
WHERE  Poblacion / @PoblacionMundial * 100 >= 1
ORDER BY Poblacion DESC;
```

# Definición de un procedimiento

Un programa almacenado en la base de datos, también conocido como programa, rutina o módulo almacenado (stored program, routine or module) es un conjunto de comandos SQL que pueden almacenarse y ejecutarse en el servidor de bases de datos.

**Procedimiento** almacenado: es programa que se ejecuta cuando se le llama que puede aceptar múltiples parámetros de entrada y salida.

**Función** almacenada: igual que un procedimiento, pero su devuelve un escalar (único valor) que sustituye al nombre de la función.

**Disparador**: programa almacenado que se ejecuta en respuesta a determinada actividad en la BD, normalmente suele ser una orden de INSERT, UPDATE o DELETE. Se usa para validación de datos o para automatizar la normalización.

```
DROP PROCEDURE IF EXISTS P1;
DELIMITER //
CREATE PROCEDURE P1(IN Cadena1 VARCHAR(50))
BEGIN
        DECLARE Cadena2  VARCHAR(50) DEFAULT 'Hola';

    SET Cadena2:= 'Hola';
    SELECT CONCAT(Cadena2, ' ', Cadena1);
END//
DELIMITER ;
CALL P1('mundo');
```

# Variables

Sólo pueden estar dentro de un programa almacenado
Hay que declararlas al principio del programa

# Definición de una función

```
CREATE FUNCTION UnoDeDos (IntA INT, IntB INT)
   RETURNS INT
BEGIN
   IF RAND()>0.6
       THEN RETURN IntA;
       ELSE RETURN IntB;
    END IF;
END//
SELECT(UnoDeDos(5, 9) + 3);
```

# Parámetros

```
[ IN | OUT | INOUT ] param_name type
```

- Si no se pone ni *IN*, ni *OUT*, ni *INOUT* el parámetro es *IN* por defecto, pero nosotros no omitiremos el *IN*
- Sólo sirven para los procedimientos. Todos los parámetros de una función son *IN*. En el caso de funciones no se puede poner nada, ni siquera *IN*

# IF

```
IF     Numero>8 THEN Ordenes;
ELSEIF Numero<8 THEN Ordenes;
ELSEIF Numero=8 THEN Ordenes;
ELSE Ordenes;
END IF;
```

# CASE

```
CASE IntA
    WHEN 1 THEN Ordenes;
    WHEN 2 THEN Ordenes;
    ELSE Ordenes;
END CASE;

CASE
    WHEN IntA=1 THEN Ordenes;
    WHEN IntA=2 THEN Ordenes;
    ELSE Ordenes;
END CASE;
```

# Bucle LOOP

```
Bucle: LOOP
    IF Contador>=Numero THEN LEAVE Bucle; END IF;
    SELECT 'Hola ola' AS 'Resultado';
    SET Contador= Contador+1;
END LOOP Bucle;
```

# Bucle REPEAT-UNTIL

```
REPEAT
    SELECT 'Hola ola' AS 'Resultado';
    SET Contador= Contador+1;
UNTIL Contador >= Numero OR Numero IS NULL END REPEAT;
```

# Bucle WHILE

```
WHILE Contador<Numero AND Numero IS NOT NULL DO
    SELECT 'Hola ola' AS 'Resultado';
    SET Contador= Contador+1;
END WHILE;
```

# Etiquetas

- Se permiten etiquetas en los bloques *BEGIN ... END* y en las ordenes *LOOP*, *REPEAT* y *WHILE*.
- Con las etiquetas podemos usar las órdenes *ITERATE* O *LEAVE*.
- *ITERATE* sólo puede usarse con bucles *LOOP*, *REPEAT* y *WHILE*.

```
[begin_label:] BEGIN
[statement_list]
END [end_label]

[begin_label:] LOOP
statement_list
END LOOP [end_label]

[begin_label:] REPEAT
statement_list
UNTIL search_condition
END REPEAT [end_label]

[begin_label:] WHILE search_condition DO
statement_list
END WHILE [end_label]
```

# Procedimiento con etiqueta

```
CREATE FUNCTION NPI(n INT)
RETURNS INT
BEGIN
    etiqueta: LOOP
        SET n = n + 2;
        IF n < 10 THEN
            SET n = n + 1;
            ITERATE etiqueta;
        END IF;
        IF n > 20 THEN
            SET n = n + 2;
            LEAVE etiqueta;
        END IF;
    END LOOP etiqueta;
    RETURN n;
END;
SELECT(NPI(10));
```

# Llamadas a procedimientos y funciones

```
DROP PROCEDURE IF EXISTS PruebaProc;
DELIMITER //
CREATE PROCEDURE PruebaProc (OUT X INT)
BEGIN
   SET X= SIN(PI()*RAND());
END//
DELIMITER ;

CALL PruebaProc(@A);
SELECT @A*2 AS 'Resultado';

DROP FUNCTION IF EXISTS PruebaFunc;
DELIMITER //
CREATE FUNCTION PruebaFunc ()
   RETURNS INT
BEGIN
   RETURN SIN(PI()*RAND());
END//
DELIMITER ;

SELECT PruebaFunc()*2 AS Resultado;
```