Procedimientos almacenados y funciones - Implementación

Variables locales: Declaración

Una variable permite almacenar datos temporalmente.

Las variables se declaran con la siguiente sintaxis:

```
DECLARE nombre_variable tipo [ DEFAULT valor_por_defecto ]
```

Ejemplos:

```
DECLARE nombre VARCHAR(100);
```

```
DECLARE nombre VARCHAR(100) DEFAULT 'Javier Lopez Gonzalez';
```

Variables locales: Actualización

- Para actualizar una variable se utiliza SET:

```
SET nombre =  'Javier Lopez Gonzalez';
```

- También se pueden introducir los datos de las columnas de una consulta utilizando INTO:

```
SELECT columna[,...] INTO var_nombre[,...] FROM tabla;
```

Ejemplo:

```
SELECT alu_nombre INTO nombre FROM alumnos WHERE alu_codalu=1;
```

Variables definidas por el usuario

Se puede almacenar un valor en una variable definida por el usuario en una declaración y referirse a él más adelante en otra declaración. Esto le permite pasar valores de una declaración a otra.

Las variables se declaran con la siguiente sintaxis:

```
SET @var_name = expr [, @var_name = expr] ...
```

Ejemplos:

```
SET @nombre = ‘Federico’;
```

```
SET @nombre = ‘Federico’, @edad=54, @provincia=’Albacete’;
```

Variables definidas por el usuario: Ejemplo

```
CREATE PROCEDURE ejemplo_variables
()
BEGIN
   DECLARE variable INT DEFAULT 1;
   SET variable = variable + 1;
   SET @variable = @variable + 1;
   SELECT variable, @variable;
END;
```

```
/* Variable definida por el usuario */
SET @variable = 1;

CALL ejemplo_variables();
```

variable	@variable
2	2

```
CALL ejemplo_variables();
```

variable	@variable
2	3

```
CALL ejemplo_variables();
```

variable	@variable
2	4

Sentencia compuesta BEGIN ... END

Una sentencia compuesta puede contener múltiples sentencias, encerradas por las palabras BEGIN y END.

Cada sentencia dentro de BEGIN ... END debe terminar con un punto y coma (;) como delimitador de sentencias.

La sintaxis BEGIN ... END se utiliza para escribir sentencias compuestas que pueden aparecer por ejemplo en el interior de procedimientos almacenados, handlers, triggers...

Control de flujo: Sentencia IF

La sentencia IF evalúa una condición y realiza una acción si es verdadera u otra en caso contrario:

IF condición THEN realiza esta acción

[ELSEIF condición THEN realiza esta acción] ...

[ELSE realiza esta acción]

END IF

Se utiliza ELSEIF para realizar sucesivas condiciones salvo para la última, que se utiliza ELSE.

Control de flujo: Ejemplo IF

A continuación, se muestra un ejemplo sencillo:

```
IF valor1<valor2 THEN select ‘Valor1 menor que Valor2’
   ELSEIF valor1=valor2 THEN select ‘Valor1 igual que Valor2’
   ELSE select ‘Valor1 mayor que Valor2’
END IF
```

Control de flujo: Sentencia CASE

La sentencia CASE evalúa el valor de una variable y realiza una acción correspondiente:

```
CASE variable

    WHEN valor1 THEN realiza esta acción

    WHEN valor2 THEN realiza esta acción

    ELSE realiza esta acción

END CASE;
```

Se utiliza WHEN *valor* THEN para realizar la acción correspondiente cuando la variable tiene ese valor. Y para el resto de valores se utiliza ELSE.

Control de flujo: Ejemplo CASE

A continuación, se muestra un ejemplo sencillo:

```
CASE dorsal
   WHEN 1 THEN select ‘Portero’
   WHEN 3 THEN select ‘Defensa’
   WHEN 10 THEN  select ‘Delantero’
   ELSE  select ‘Suplente’
END CASE;
```

Control de flujo: Sentencia WHILE

La sentencia WHILE permite la realización de un número indeterminado de veces las acciones que lo contienen, **mientras** se cumpla la condición:

```
WHILE condición DO
        realiza esta acción
END WHILE;
```

Ejemplo:

```
WHILE valor1<valor2 DO
    set valor1=valor1+1;
END WHILE;
```

Control de flujo: Sentencia REPEAT

La sentencia REPEAT permite la realización de un número indeterminado de veces las acciones que lo contienen, **hasta** que se cumpla la condición. La gran diferencia con respecto a WHILE es que con REPEAT las acciones que lo contienen se realizan al menos una vez.

```
REPEAT
        realiza esta acción
UNTIL condición
END REPEAT;
```

Ejemplo:

```
REPEAT
    set valor1=valor1+1;
UNTIL valor1>valor2
END REPEAT;
```