A photograph showing the back of a person with long blonde hair working at a computer. The monitor displays a dark interface with multiple windows open, showing code or database structures. A second monitor is visible in the background displaying a greenish landscape. A white coffee cup sits on the desk to the right.

Implementación y gestión de bases de datos

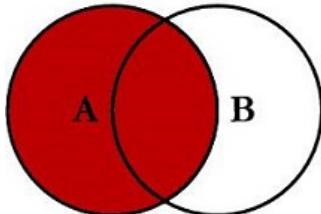
Consultas combinadas DML

Consultas combinadas DML

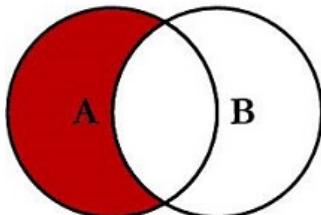
Figura 1

Comandos JONS en SQL

SQL JOINS

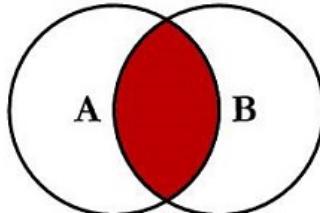


```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

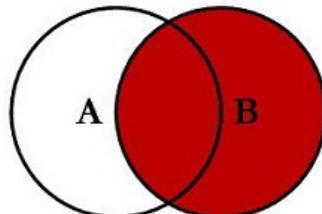


```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

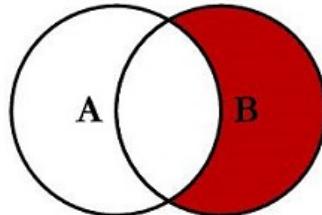
```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



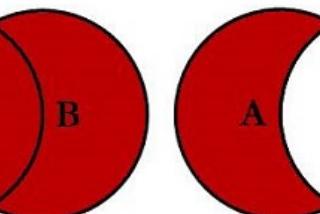
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

Elementos del lenguaje

Variables

Las variables son muy importantes y le adiciona un valor agregado en los scripts de SQL, con el objetivo de crear variables como por ejemplo el inicio de un bucle (ciclo). Las variables permiten guardar un valor y poder recuperarlo más adelante para utilizarlo en otras sentencias.

Podemos usar la instrucción DECLARE para indicar o declarar una o más variables.

DECLARE @variableejemplo INT

Para asignar un valor a la variable declarada se debe colocar el signo igual (=) y luego el valor correspondiente, ejemplo:

```
SELECT @nombre = nombre FROM empleados WHERE cargo='operario';
```

Para consultar el valor correspondiente de una variable: SELECT @nombre;

Estructuras de control

Las estructuras de control permiten seleccionar la manera como queremos ejecutar las diferentes instrucciones. Existen 3 diferentes estructuras de control:

Secuencial, las instrucciones se ejecutan una después de otra de forma sucesiva

Alternativa, las instrucciones tienen dos caminos por donde se pueden ejecutar dependiendo de una condición.

Repetitiva, las instrucciones se ejecutan varias veces dependiendo de una condición respectivamente.

Condicionales – IF

La estructura condicional IF permite evaluar una expresión por medio de una condición para obtener un resultado booleano (si o no), y ejecutar las instrucciones contenidas dentro del bloque. La sintaxis es la siguiente:

```
IF(expresión1)
BEGIN
Instrucción 1;
END
ELSE IF(expresión2)
BEGIN
Instrucción 2;
END
ELSE
BEGIN
Instrucción 3;
END
```

Ejemplo:

```
DECLARE @totalCompras INT
SELECT @totalCompras = COUNT(*) FROM compras
IF @totalCompras > 50
PRINT 'existen más de 50 productos'
ELSE
PRINT 'existen menos de 50 productos'
```

Estructura condicional CASE

Este tipo de estructura condicional permite evaluar una determinada expresión y devolver un valor u otro. La sintaxis es la siguiente:

```
CASE <expresion>
WHEN <valor_expresion> THEN <valor_retorno>
WHEN <valor_expresion> THEN <valor_retorno>
ELSE <valor_retorno> -- valor por defecto
END
```

Ejemplo:

```
SELECT precio =
CASE
WHEN precio_unidad IS NULL THEN 'Desconocido'
WHEN precio_unidad < 10000 THEN 'Precio bajo'
WHEN precio_unidad > 10000 THEN 'Precio alto'
ELSE 'Vale exactamente 10000'
END, nombre_producto
FROM productos
```

Bucle WHILE

Este tipo de bucle o ciclo se repite cada una de las instrucciones mientras la expresión se evalúa como verdadera. La sintaxis es la siguiente:

```
DECLARE <expresion>
BEGIN
Intrucciones...
END
```

Ejemplo:

```
DECLARE @Contador INT
SET @Contador = 5
WHILE (@Contador < 0)
BEGIN
Print '@Contador =' + CONVERT(NVARCHAR, @Contador)
SET @Contador = @Contador + 1
END
Bucle FOR
```

El funcionamiento es muy similar al bucle WHILE, lo único es que tiene un inicio y un fin de valor para su respectiva iteración. La sintaxis es la siguiente:

```
FOR variable_contador IN [REVERSE] valor_inicial ... valor_final
LOOP
{instrucciones...}
END LOOP;
```

Ejemplo:

```
BEGIN
FOR i IN 1..10
LOOP
DBMS.OUTPUT.PUT_LINE(i);
END LOOP;
END;
```

Bucle FOREACH

Este tipo de bucle es similar al bucle FOR, en cada iteración realiza el cálculo de un conjunto de datos y su retorno es un determinado conjunto de datos. La sintaxis es la siguiente:

FOREACH id_1 IN set_1 [, id_n IN set_n] RETURN(expression)

El id es un identificador para el elemento que se va a recorrer en cada iteración

SET es un conjunto de datos, de cualquier tipo de datos del conjunto

Debe contener RETURN, para expresar el respectivo retorno de la expresión.

Ejemplo:

```
RETURN resultado AS
SELECT
Codigo.x AS x,
FOREACH x IN {1, 2}, y IN {3, 4} RETURN (resultado.x + Codigo.y) AS valores
FROM Codigo
```

Manejando Excepciones.

Esta clase de excepciones en el lenguaje SQL, se implementa para validar los posibles errores que puedan ocurrir en tiempos de ejecución de la sentencia SQL. Se utiliza el comando EXCEPTION.

Normalmente en un bloque de SQL expresado en forma declarativa se conserva el siguiente orden:

Comienza con la palabra clave DECLARE

Una sección ejecutable dentro de un bloque BEGIN y termina en END

Para el manejo de excepciones se incluye otro bloque opcional, comienza con la palabra EXCEPTION, siempre forma parte del último bloque de la expresión declarativa

La sintaxis es la siguiente:

```
SET SERVEROUTPUT ON
DECLARE
    apellido VARCHAR2(15);
BEGIN
    SELECT segundo_nombre INTO apellido
    FROM empleados
    WHERE primer_nombre = 'John';
    DBMS_OUTPUT.PUT_LINE ('El Apellido de John es:' || apellido);
EXCEPTION
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE ('Tu consulta retorna más de un registro. Usa un Cursor.');
END;
```