Consultas Simples

El corazón o poder del Lenguaje SQL es el poder hacer consultas de cualquier tipo a la base de datos en forma no procedural. La sentencia **SELECT** es muy poderosa y ampliamente rica en sus cláusulas y variantes permitiendo la capacidad de atender en poco tiempo a consultas complejas sobre la base de datos. Está en el desarrollador de aplicaciones conocerlo a profundidad para explotar las bondades y virtudes.

La sentencia SELECT, obtiene filas de la base de datos y permite realizar la selección de una o varias filas o columnas de una o varias tablas.

La sintaxis completa de la instrucción SELECT es compleja, pero la voy resumir como sigue (los corchetes cuadrados indican que la cláusula no es obligatoria):

SELECT nombres de las columnas

[INTO nueva Tabla destino para resultados del select_]

FROM origenTabla

[WHERE condición de Búsqueda]

[GROUP BY nombres de columnas por la cual Agrupar]

[HAVING condiciónBúsqueda para Group By]

[ORDER BY nombre de columnas [ASC | DESC]]

Veamos cuales son las funciones de cada una de las cláusulas de la sentencia SELECT.

La cláusula SELECT: Se usa para listar las columnas de las tablas que se desean ver en el resultado de una consulta. Además de las columnas se pueden listar columnas a calcular por el SQL cuando actué la sentencia. Esta cláusula no puede omitirse.

La cláusula FROM: Lista las tablas que deben ser analizadas en la evaluación de la expresión de la cláusula WHERE y de donde se listarán las columnas enunciadas en el SELECT. Esta cláusula no puede omitirse.

La cláusula WHERE: establece criterios de selección de ciertas filas en el resultado de la consulta gracias a las condiciones de búsqueda. Si no se requiere condiciones de búsqueda puede omitirse y el resultado de la consulta serán todas las filas de las tablas enunciadas en el FROM.

La cláusula GROUP BY: especifica una consulta sumaria. En vez de producir una fila de resultados por cada fila de datos de la base de datos, una consulta sumaria agrupa todas las filas similares y luego produce una fila sumaria de resultados para cada grupo de los nombres de columnas enunciado en esta cláusula. En otras palabras, esta cláusula permitirá agrupar un conjunto de columnas con valores repetidos y utilizar las funciones de agregación sobre las columnas con valores no repetidas. Esta cláusula puede omitirse.

La cláusula HAVING: le dice al SQL que incluya sólo ciertos grupos producidos por la cláusula GROUP BY en los resultados de la consulta. Al igual que la cláusula WHERE, utiliza una condición de búsqueda para especificar los grupos deseados. La cláusula HAVING es la encargada de condicionar la selección de los grupos. Esta cláusula puede omitirse.

La cláusula ORDER BY: permitirá establecer la columna o columnas sobre las cuales las filas resultantes de la consulta deberán ser ordenadas. Esta cláusula puede omitirse

Sentencia SELECT-FROM

El utilizar la sentencia SELECT, con estas dos cláusulas SELECT - FROM, muestra como resultado todas las filas existentes en las tablas especificadas en el FROM.

Ejemplo # 1: Seleccionar todas las columnas y filas de la tabla Alumnos

```
SELECT * FROM Alumnos
```

Ejemplo # 2: Seleccionar columnas: nombre y apellidos de la tabla Alumnos

```
SELECT nombre, apellidos FROM Alumnos
```

Ejemplo # 3: Este ejemplo selecciona las columnas y las muestra con el título especificado en AS, es decir con un alias. Al campo email lo muestra como Correo Electronico y codigocurso lo muestra con el nombre especificado en el AS, en este caso Numero Curso. Esto permite mostrar una columna con el nombre que queramos.

```
SELECT email AS 'Correo Electronico', codigocurso AS 'Numero Curso' FROM Alumnos
```

La cláusula AS puede omitirse y el resultado es el mismo.

```
SELECT email 'Correo Electronico', codigocurso 'Numero Curso' FROM Alumnos
```

SENTENCIA DE FILAS DUPLICADAS (DISTINCT)

Si una consulta incluye la llave primaria (pk) de una tabla en su lista de selección, entonces cada fila de resultados será única (ya que la llave primaria (pk) tiene un valor diferente en cada fila). Si no se incluye la llave primaria en los resultados, pueden producirse filas duplicadas. Veamos el siguiente ejemplo,

Ejemplo # 5: Seleccionar el código de curso sin usar la palabra reservada DISTINCT.

SELECT codigocurso FROM alumnos ORDER BY codigo

Ejemplo # 6: Seleccionar el código de curso utilizando la palabra reservada DISTINCT. (Se obtienen menos resultados)

SELECT DISTINCT codigocurso FROM alumnos ORDER BY codigo

Columnas Calculadas

Además de las columnas cuyos valores serán introducidos a la base de datos a través de la sentencia INSERT, una consulta SQL puede incluir en su cláusula SELECT columnas calculadas cuyo valor se calculan a partir de los valores de las otras columnas almacenadas en las tablas. Estas columnas, son especie de una columna virtual pues no existen físicamente en la tabla y sus valores calculados corresponden a los valores por fila.

Supongamos que tenemos una tabla CLIENTE con tres columnas que almacenan saldos con los nombres SALDO_0_30, SALDO_31_60 y SALDO_61_90

Ejemplo # 7: Determinar los clientes con saldo.

```
SELECT
(SALDO_0_30 +SALDO_31_60 + SALDO_61_90) as "Saldo del
Cliente"  FROM CLIENTE
WHERE (SALDO_0_30 +SALDO_31_60 + SALDO_61_90) <> 0
```

Condiciones de Búsqueda (=, <>, >, <, >=, <=, BETWEEN, IN, LIKE, IS NULL, compuestas (AND, OR, NOT))

SQL usa las conectivas lógicas AND, OR y NOT en la cláusula WHERE. Los operandos de las conectivas lógicas pueden ser expresiones que contengan los operadores de comparación <,<=,>, > =, = y <>. SQL permite usar los operadores

de comparación para comparar cadenas y expresiones aritméticas, así como tipos especiales, tales como el tipo fecha.

Condiciones de Búsqueda (=, <>, >, <, >=, <=):

Ejemplo # 8: Seleccionar a los alumnos cuyo número de código está contenido entre el rango 5 y 10.

```
SELECT *  FROM alumnos
WHERE codigo >= 5 AND codigo <= 10
```

Se obtiene el mismo resultado utilizando la palabra reservada BETWEEN. Este especifica que un valor sea menor o igual que un valor y mayor o igual que otro valor.

Condiciones de Búsqueda con BETWEEN:

Ejemplo # 9: Seleccionar a los alumnos cuyo número de código está contenido entre el rango 5 y 10.

```
SELECT * FROM Alumnos WHERE codigo BETWEEN 5 AND 10
```

Condiciones de Búsqueda Compuestas con AND, OR y NOT:

Ejemplo # 10: Se puede combinar las conectivas lógicas AND, OR or y NOT y filtrar mayor la selección en la consulta. Por ejemplo al seleccionar a los alumnos cuyo código está contenido entre el rango 5 y 10 y que no están inscritos en el curso con codigocurso 1.

```
SELECT * FROM Alumnos WHERE codigo BETWEEN 5
AND 10  AND NOT codigocurso = 1
```

Ejemplo # 11: Utilizando el operador OR podemos buscar un alumno cuyo código dudamos si es 12 o 21.

```
SELECT * FROM Alumnos WHERE codigo= 12 OR codigo = 21
```

Ejemplo # 12: Para seleccionar los alumnos cuyo código es mayor de 8 y su codigocurso superior a 3.

```
SELECT * FROM Alumnos WHERE codigo>8  AND codigocurso >
3
```

Condiciones de Búsqueda con LIKE:

Ejemplo # 13: Seleccionar a todos los alumnos que contengan la letra "H" dentro de su nombre.

```
SELECT *FROM Alumnos WHERE nombre LIKE '%H%'
```

Condiciones de Búsqueda con IS NULL:

Los valores de NULL crean una lógica trivaluada para las condiciones de búsqueda en SQL. Para una fila determinada, el resultado de una condición de búsqueda puede ser TRUE o FALSE, o puede ser NULL debido a que una de las columnas
utilizadas en la evaluación de la condición de búsqueda contenga un valor NULL.

A veces es útil comprobar explícitamente los valores NULL en una condición de búsqueda y manejarlas directamente.

```
SELECT *
FROM nombre_tabla
WHERE campo_tabla IS NULL
```

Ordenación de resultados de Consulta (cláusula ORDER BY)

Al igual que las filas de una tabla en la base de datos las filas de los resultados de una consulta no están dispuestas en ningún orden particular. Existen situaciones en la que es necesario ver la información en un orden en especial, como en orden alfabético (ASC, ascendente) o ver a las cifras de mayor a menor (DESC, descendente). Se puede pedir a SQL que ordene los resultados de una consulta incluyendo la cláusula ORDER BY en la sentencia SELECT.

Ejemplo # 14: Para buscar la información de los alumnos por orden de su apellido, la sentencia select con la cláusula ORDER BY seria la siguiente:

```
SELECT *
FROM alumnos
ORDER BY apellido
```

Consultas a Múltiples Tablas

Generalmente el poder de la sentencia SELECT se basa en su capacidad de poder en una sola sentencia consultar múltiples tablas simultáneamente. Esta operación también se le llama JOIN y es posible gracias a que existen columnas de conexión, es decir atributos de asociación comunes en las tablas.

Existen dos formas de sintaxis permitidas para la escritura de la sentencia SELECT para la unión de tablas relacionadas. Estas formas son las siguientes:

FORMA 1:

```
SELECT a.nombre, a.apellidos, a.email, a.codigocurso, c.nombre

FROM alumnos a ,cursos c

WHERE a.codigocurso = c.codigo
```

FORMA 2:

```
SELECT a.nombre, a.apellidos, a.email, a.codigocurso, c.nombre

FROM Alumnos a INNER JOIN Cursos c ON c.codigo =

a.codigocurso
```

Note que las columnas comunes en ambas tablas han de calificarse con el nombre de la tabla que pertenecen para evitar errores de ambigüedad.

La Forma 2, tiene la ventaja de liberar a la cláusula WHERE y dejar esta para filtros específicos sobre las filas resultantes de la reunión de tablas.