INFORMATORIO

BASE DE DATOS

Lenguaje SQL: Criterios de selección

Subsecretaría de Empleo
Chaco Gobierno de todos

Ministerio de Producción, Industria y Empleo
Chaco Gobierno de todos

CHACO
Gobierno de todos

Parte 3: Búsqueda y selección de datos en SQL

Estudiamos a fondo todo lo relacionado con la sentencia select dentro del lenguaja SQL.

3.6.- Criterios de selección en SQL

Estudiaremos las posibilidades de filtrar los registros con el fin de recuperar solamente aquellos que cumplan unas condiciones preestablecidas.

Antes de comenzar el desarrollo de este apartado hay que recalcar tres detalles de vital importancia. El **primero** de ellos es que cada vez que se desee establecer una condición referida a un campo de texto la condición de búsqueda debe ir encerrada entre comillas simples; la segunda es que no es posible establecer condiciones de búsqueda en los campos memo y; la tercera y última hace referencia a las fechas. A día de hoy no he sido capaz de encontrar una sintaxis que funcione en todos los sistemas, por lo que se hace necesario particularizarlas según el motor de base de datos:

Banco de Datos	**Sintaxis**
SQL-SERVER	Fecha = #mm-dd-aaaa#
ORACLE	Fecha = to_date('YYYYDDMM','aaaammdd',)
ACCESS	Fecha = #mm-dd-aaaa#

Referente a los valores lógicos True o False cabe destacar que no son reconocidos en ORACLE, ni en este sistema de bases de datos ni en SQL-SERVER existen los campos de tipo "SI/NO" de ACCESS; en estos sistemas se utilizan los campos BIT que permiten almacenar valores de 0 ó 1. Internamente, ACCESS, almacena en estos campos valores de 0 ó -1, así que todo se complica bastante, pero aprovechando la coincidencia del 0 para los valores FALSE, se puede utilizar la

sintaxis siguiente que funciona en todos los casos: si se desea saber si el campo es falso "... CAMPO = 0" y para saber los verdaderos "CAMPO <> 0".

3.6.1.- Operadores Lógicos

Los operadores lógicos mas comunes en SQL son: AND, OR, Is, Not. A excepción de los dos últimos todos poseen la siguiente sintaxis:

<expresión1> operador <expresión2>

En donde expresión1 y expresión2 son las condiciones a evaluar, el resultado de la operación varía en función del operador lógico.
La tabla adjunta muestra los diferentes posibles resultados para AND y OR:

<expresión1>	Operador	<expresión2>	Resultado
Verdad	AND	Falso	Falso
Verdad	AND	Verdad	Verdad
Falso	AND	Verdad	Falso
Falso	AND	Falso	Falso
Verdad	OR	Falso	Verdad
Verdad	OR	Verdad	Verdad
Falso	OR	Verdad	Verdad
Falso	OR	Falso	Falso

Si a cualquiera de las anteriores condiciones le anteponemos el operador NOT el resultado de la operación será el contrario al devuelto sin el operador NOT.

El último operador denominado Is se emplea para comparar dos variables de tipo objeto <Objeto1> Is <Objeto2>. este operador devuelve verdad si los dos objetos son iguales.

```
SELECT *
FROM
    Empleados
WHERE
    Edad > 25 AND Edad < 50

SELECT *
FROM
    Empleados
WHERE
        (Edad > 25 AND Edad < 50)
 OR

Sueldo = 100

SELECT *
FROM Empleados
WHERE NOT Estado = 'Soltero'

SELECT *
 FROM Empleados
WHERE (Sueldo >100 AND Sueldo < 500)
        OR
       (Provincia = 'Madrid' AND Estado = 'Casado')
```

3.6.2.- Intervalos de Valores

Para indicar que deseamos recuperar los registros según el intervalo de valores de un campo emplearemos el operador
Between cuya sintaxis es:

```
campo [Not] Between valor1 And valor2 (la condición Not es opcional)
```

En este caso la consulta devolvería los registros que contengan en "campo" un valor incluido en el intervalo valor1, valor2 (ambos inclusive). Si anteponemos la condición Not devolverá aquellos valores no incluidos en el intervalo.

```
SELECT *
FROM
    Pedidos
WHERE
    CodPostal Between 28000 And 28999
(Devuelve los pedidos realizados en la provincia de Madrid)
```

3.6.3.- El Operador Like

Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL. Su sintaxis es:

```
expresión Like modelo
```

En donde expresión es una cadena modelo o campo contra el que se compara expresión. Se puede utilizar el operador Like para encontrar valores en los campos que coincidan con el modelo especificado. Por modelo puede especificar un valor completo (Ana María), o se puede utilizar una cadena de caracteres comodín como los reconocidos por el sistema operativo para encontrar un rango de valores (Like An*).

El operador Like se puede utilizar en una expresión para comparar un valor de un campo con una expresión de cadena. Por ejemplo, si introduce Like C* en una consulta SQL, la consulta devuelve todos los valores de campo que comiencen por la letra C. En una consulta con parámetros, puede hacer que el usuario escriba el modelo que se va a utilizar.

El ejemplo siguiente devuelve los datos que comienzan con la letra P seguido de cualquier letra entre A y F y de tres dígitos:

```
Like 'P[A-F]###'
```

Este ejemplo devuelve los campos cuyo contenido empiece con una letra de la A a la D seguidas de cualquier cadena.

```
Like '[A-D]*'
```

En la tabla siguiente se muestra cómo utilizar el operador Like en SQL Server para comprobar expresiones con diferentes modelos.

Ejemplo	Descripción
LIKE 'A%'	Todo lo que comience por A
LIKE '_NG'	Todo lo que comience por cualquier carácter y luego siga NG
LIKE '[AF]%'	Todo lo que comience por A ó F
LIKE '[A-F]%'	Todo lo que comience por cualquier letra comprendida entre la A y la F
LIKE '[A^B]%'	Todo lo que comience por A y la segunda letra no sea una B

En determinado motores de bases de datos, esta cláusula, no reconoce el asterisco como carácter comodín y hay que sustituirlo por el carácter tanto por ciento (%).

3.6.4.- El Operador In

Este operador devuelve aquellos registros cuyo campo indicado coincide con alguno de los en una lista. Su sintaxis es:

```
expresión [Not] In(valor1, valor2, . . .)

SELECT *
FROM
    Pedidos
WHERE

    Provincia In ('Madrid', 'Barcelona', 'Sevilla')
```

3.6.5.- La cláusula WHERE

La cláusula WHERE puede usarse para determinar qué registros de las tablas enumeradas en la cláusula FROM aparecerán en los resultados de la instrucción SELECT. Después de escribir esta cláusula se deben especificar las condiciones expuestas en los apartados anteriores. Si no se emplea esta cláusula, la consulta devolverá todas las filas de la tabla. WHERE es opcional, pero cuando aparece debe ir a continuación de FROM.

```
SELECT
    Apellidos, Salario
FROM
    Empleados
WHERE
    Salario = 21000
```

```
SELECT
   IdProducto, Existencias
FROM
   Productos
WHERE
    Existencias <= NuevoPedido
```

```
SELECT *
FROM
  Pedidos
WHERE
  FechaEnvio = #05-30-1994#
```

```
SELECT
    Apellidos, Nombre
FROM
    Empleados
WHERE
   Apellidos = 'King'
```

```
SELECT
    Apellidos, Nombre
FROM
  Empleados
WHERE
   Apellidos Like 'S*'
```

```
SELECT
  Apellidos, Salario
FROM
   Empleados
WHERE
   Salario Between 200 And 300
```

```
SELECT
    Apellidos, Salario
FROM
    Empleados
WHERE
 Apellidos Between 'Lon' And 'Tol'
```

```
SELECT
    IdPedido, FechaPedido
FROM
    Pedidos
WHERE
FechaPedido Between #01-01-1994# And #12-31-1994#
```

```
SELECT Apellidos, Nombre, Ciudad
FROM Empleados
WHERE
  Ciudad In ('Sevilla', 'Los Angeles', 'Barcelona')
```

3.7.- Criterios de selección en SQL II

Seguimos con el group by, avg, sum y con el compute de sql-server.

Combina los registros con valores idénticos, en la lista de campos especificados, en un único registro. Para cada registro se crea un valor sumario si se incluye una función SQL agregada, como por ejemplo Sum o Count, en la instrucción SELECT. Su sintaxis es:

```
SELECT campos FROM tabla WHERE criterio GROUP BY campos del grupo
```

GROUP BY es opcional. Los valores de resumen se omiten si no existe una función SQL agregada en la instrucción SELECT. Los valores Null en los campos GROUP BY se agrupan y no se omiten. No obstante, los valores Null no se evalúan en ninguna de las funciones SQL agregadas.

Se utiliza la cláusula WHERE para excluir aquellas filas que no desea agrupar, y la cláusula HAVING para filtrar los registros una vez agrupados.
A menos que contenga un dato Memo u Objeto OLE, un campo de la lista de campos GROUP BY puede referirse a cualquier campo de las tablas que aparecen en la cláusula FROM, incluso si el campo no esta incluido en la instrucción SELECT, siempre y cuando la instrucción SELECT incluya al menos una función SQL agregada.

Todos los campos de la lista de campos de SELECT deben o bien incluirse en la cláusula GROUP BY o como argumentos de una función SQL agregada.

```
SELECT
    IdFamilia, Sum(Stock) AS StockActual
FROM
     Productos
GROUP BY

     IdFamilia
```

Una vez que GROUP BY ha combinado los registros, HAVING muestra cualquier registro agrupado por la cláusula GROUP BY que satisfaga las condiciones de la cláusula HAVING.

HAVING es similar a WHERE, determina qué registros se seleccionan. Una vez que los registros se han agrupado utilizando GROUP BY, HAVING determina cuales de ellos se van a mostrar.

```
SELECT
    IdFamilia, Sum(Stock) AS StockActual
FROM
     Productos
GROUP BY
     IdFamilia
HAVING
     StockActual > 100
    AND

    NombreProducto Like BOS*
```

3.7.1.- AVG

Calcula la media aritmética de un conjunto de valores contenidos en un campo especificado de una consulta. Su sintaxis es la siguiente

```
Avg(expr)
```

En donde expr representa el campo que contiene los datos numéricos para los que se desea calcular la media o una expresión que realiza un cálculo utilizando los datos de dicho campo. La media calculada por Avg es la media aritmética (la suma de los valores dividido por el número de valores). La función Avg no incluye ningún campo Null en el cálculo.

```
SELECT
    Avg(Gastos) AS Promedio
FROM
    Pedidos
WHERE
    Gastos > 100
```

3.7.2.- Count

Calcula el número de registros devueltos por una consulta. Su sintaxis es la siguiente

```
Count(expr)
```

En donde expr contiene el nombre del campo que desea contar. Los operandos de expr pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL). Puede contar cualquier tipo de datos incluso texto.

Aunque expr puede realizar un cálculo sobre un campo, Count simplemente cuenta el número de registros sin tener en cuenta qué valores se almacenan en los registros. La función Count no cuenta los registros que tienen campos null a menos que expr sea el carácter comodín asterisco (*). Si utiliza un asterisco, Count calcula el número total de registros, incluyendo aquellos que contienen campos null. Count(*) es considerablemente más rápida que Count(Campo). No se debe poner el asterisco entre dobles comillas ('*').

```
SELECT
   Count(*) AS Total
FROM

   Pedidos
```

Si expr identifica a múltiples campos, la función Count cuenta un registro sólo si al menos uno de los campos no es Null. Si todos los campos especificados son Null, no se cuenta el registro. Hay que separar los nombres de los campos con ampersand (&).

```
SELECT
   Count(FechaEnvío & Transporte) AS Total
FROM
    Pedidos
```

Podemos hacer que el gestor cuente los datos diferentes de un determinado campo

```
SELECT
    Count(DISTINCT Localidad) AS Total
FROM

    Pedidos
```

3.7.3.- Max, Min

Devuelven el mínimo o el máximo de un conjunto de valores contenidos en un campo especifico de una consulta. Su sintaxis es:

```
Min(expr)

Max(expr)
```

En donde expr es el campo sobre el que se desea realizar el cálculo. Expr pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL).

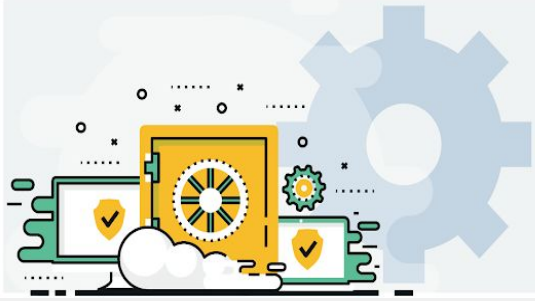
```
SELECT

Min(Gastos) AS ElMin

FROM
    Pedidos
WHERE
    Pais = 'España'

SELECT
    Max(Gastos) AS ElMax
FROM
    Pedidos
WHERE
     Pais = 'España'
```

INFORMATORIO



3.7.4.- Sum

Devuelve la suma del conjunto de valores contenido en un campo especifico de una consulta. Su sintaxis es:

```
Sum(expr)
```

En donde expr representa el nombre del campo que contiene los datos que desean sumarse o una expresión que realiza un cálculo utilizando los datos de dichos campos. Los operandos de expr pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL).

```
SELECT
    Sum(PrecioUnidad * Cantidad) AS Total
FROM
    DetallePedido
```