INFORMATORIO

BASE DE DATOS

Lenguaje SQL: Selección de datos

Subsecretaría de Empleo
Chaco Gobierno de todos

Ministerio de Producción, Industria y Empleo
Chaco Gobierno de todos

CHACO
Gobierno de todos

INFORMATORIO

Parte 3: Búsqueda y selección de datos en SQL

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

Utilizar esta tabla de ejemplo:

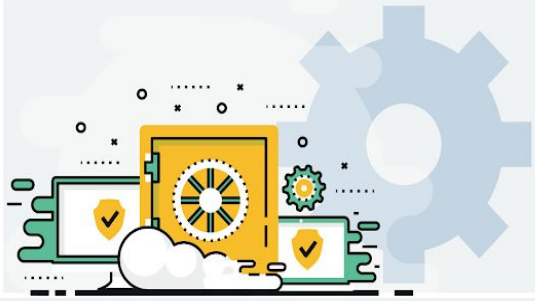
CLIENTES
nombre VARCHAR(30)
apellidos VARCHAR(30)
direccion VARCHAR(50)
población VARCHAR(20)
codigopostal lNT(4)
email VARCHAR(60)
pedidos INT(3)

3.1.- Selección de tablas I

Cómo realizar selecciones eficientemente. Ejemplos prácticos.

La selección total o parcial de una tabla se lleva a cabo mediante la instrucción Select.
En dicha selección hay que especificar:

-Los campos que queremos seleccionar
-La tabla en la que hacemos la selección



En nuestra tabla modelo de clientes podríamos hacer por ejemplo una selección del nombre y dirección de los clientes con una instrucción de este tipo:

```
Select nombre, dirección From clientes
```

Si quisiésemos seleccionar todos los campos, es decir, **toda la tabla**, podríamos utilizar el comodín * del siguiente modo:

```
Select * From clientes
```

Resulta también muy útil el filtrar los registros mediante condiciones que vienen expresadas después de la **cláusula Where**.

Si quisiésemos mostrar los clientes de una determinada ciudad usaríamos una expresión como esta:

```
Select * From clientes Where poblacion Like 'Madrid'
```

Además, podríamos **ordenar los resultados** en función de uno o varios de sus campos. Para este ultimo ejemplo los podríamos ordenar por nombre así:

```
Select * From clientes Where poblacion Like 'Madrid' Order By nombre
```

Teniendo en cuenta que puede haber más de un cliente con el mismo nombre, podríamos dar un segundo criterio que podría ser el apellido:

```
Select * From clientes Where poblacion Like 'Madrid' Order By nombre, apellido
```

Si invirtiésemos el orden « nombre,apellido » por « apellido, nombre », el resultado sería distinto. Tendríamos los clientes ordenados por apellido y aquellos que tuviesen apellidos idénticos se subclasificarían por el nombre.

Es posible también **clasificar por orden inverso**. Si por ejemplo quisiésemos ver nuestros clientes por orden de pedidos realizados teniendo a los mayores en primer lugar escribiríamos algo así:

```
Select * From clientes Order By pedidos Desc
```

Una opción interesante es la de efectuar **selecciones sin coincidencia**. Si por ejemplo buscásemos el saber en qué ciudades se encuentran nuestros clientes sin necesidad de que para ello aparezca varias veces la misma ciudad usaríamos una sentencia de esta clase:

```
Select Distinct poblacion From clientes Order By poblacion
```

Así evitaríamos ver repetido Madrid tantas veces como clientes tengamos en esa población.

3.2.- Selección de tablas II

Lista de operadores y ejemplos prácticos para realizar selecciones.

Hemos querido compilar a modo de tabla ciertos operadores que pueden resultar útiles en determinados casos. Estos operadores serán utilizados después de la cláusula Where y pueden ser **combinados hábilmente mediante paréntesis** para optimizar nuestra selección a muy altos niveles.

Operadores matemáticos:

>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
<>	Distinto
=	Igual

Operadores lógicos

And
Or
Not

Otros operadores

Like	Selecciona los registros cuyo valor de campo se asemeje, no teniendo en cuenta mayúsculas y minúsculas.
In y Not In	Da un conjunto de valores para un campo para los cuales la condición de selección es (o no) valida
Is Null y Is Not Null	Selecciona aquellos registros donde el campo especificado esta (o no) vacío.
Between... And	Selecciona los registros comprendidos en un intervalo
Distinct	Selecciona los registros no coincidentes
Desc	Clasifica los registros por orden inverso

Comodines

*	Sustituye a todos los campos
%	Sustituye a cualquier cosa o nada dentro de una cadena
_	Sustituye un solo carácter dentro de una cadena

Veamos a continuación aplicaciones prácticas de estos operadores.

En esta sentencia seleccionamos todos los clientes de Madrid cuyo nombre no es Pepe. Como puede verse, empleamos **Like** en lugar de **=** simplemente para evitar inconvenientes debido al empleo o no de mayúsculas.

```
Select * From clientes Where poblacion Like 'madrid' And Not nombre Like 'Pepe'
```

Si quisiéramos recoger en una selección a los clientes de nuestra tabla cuyo **apellido comienza por A y cuyo número de pedidos esta comprendido entre 20 y 40**:

```
Select * From clientes Where apellidos like 'A%' And pedidos Between 20 And 40
```

El operador **In**, lo veremos más adelante, es muy práctico para consultas en varias tablas. Para casos en una sola tabla es empleado del siguiente modo:

```
Select * From clientes Where poblacion In ('Madrid','Barcelona','Valencia')
```

De esta forma **seleccionamos aquellos clientes que vivan en esas tres ciudades**.

3.3.- Selección de tablas III

Cómo realizar selecciones sobre varias tablas. Ejemplos prácticos basados en una aplicación de e-comercio.

Una base de datos puede ser considerada como un conjunto de tablas. Estas tablas en muchos casos están relacionadas entre ellas y se complementan unas con otras.

Refiriéndonos a nuestro clásico ejemplo de una base de datos para una aplicación de ecommerce, la tabla clientes de la que hemos estado hablando puede estar perfectamente coordinada con una tabla donde almacenamos los pedidos realizados por cada cliente. Esta tabla de pedidos puede a su vez estar conectada con una tabla donde almacenamos los datos correspondientes a cada artículo del inventario.

De este modo podríamos fácilmente **obtener informaciones contenidas** en esas tres tablas como puede ser la *designación del artículo más popular* en una determinada región donde la designación del artículo sería obtenida de la tabla de artículos, la popularidad (cantidad de veces que ese artículo ha sido vendido) vendría de la tabla de pedidos y la región estaría comprendida obviamente en la tabla clientes.

Este tipo de organización basada en múltiples tablas conectadas nos permite trabajar con tablas mucho más manejables a la vez que nos evita copiar el mismo campo en varios sitios ya que podemos acceder a él a partir de una simple llamada a la tabla que lo contiene.

Supongamos que queremos enviar un mailing a todos aquellos que hayan realizado un pedido ese mismo día. Podríamos escribir algo así:

```
Select clientes.apellidos, clientes.email
From clientes,pedidos
Where pedidos.fecha like '25/02/00' And pedidos.id_cliente= clientes.id_cliente
```

Como puede verse esta vez, después de la cláusula From, introducimos el nombre de las dos tablas de donde sacamos las informaciones. Además, el nombre de cada campo va precedido de la tabla de origen separados ambos por un punto. En los campos que poseen un nombre que solo aparece en una de las tablas, no es necesario especificar su origen aunque a la hora de leer la sentencia puede resultar más claro el precisarlo. En este caso el campo fecha podría haber sido designado como "fecha" en lugar de "pedidos.fecha".

Veamos otro ejemplo más para consolidar estos nuevos conceptos. Esta vez queremos ver el título del libro correspondiente a cada uno de los pedidos realizados:

```
Select pedidos.id_pedido, articulos.titulo
From pedidos, articulos
Where pedidos.id_articulo=articulos.id_articulo
```

En realidad la filosofía continua siendo la misma que para la consulta de una única tabla.

3.4.- Selección de tablas IV

El empleo de funciones para la explotación de los campos numéricos y otras utilidades. Ejemplos prácticos.

Además de los criterios hasta ahora explicados para realizar las consultas en tablas, SQL permite también aplicar un conjunto de funciones predefinidas. Estas funciones, aunque básicas, pueden ayudarnos en algunos momentos a expresar nuestra selección de una manera más simple sin tener que recurrir a operaciones adicionales por parte del script que estemos ejecutando.

Algunas de estas funciones son representadas en la tabla siguiente :

Sum(campo)	Calcula la suma de los registros del campo especificado
Avg(Campo)	Calcula la media de los registros del campo especificado
Count(*)	Nos proporciona el valor del numero de registros que han sido
Max(Campo)	Nos indica cual es el valor máximo del campo
Min(Campo)	Nos indica cual es el valor mínimo del campo

Dado que el campo de la función no existe en la base de datos, sino que lo estamos generando virtualmente, esto puede crear inconvenientes cuando estamos trabajando con nuestros scripts a la hora de tratar su valor y su nombre de campo. Es por ello que el valor de la **función ha de ser recuperada a partir de un alias** que nosotros especificaremos en la sentencia SQL a partir de la instrucción **AS**. La cosa podría quedar así:

```
Select Sum(total) As suma_pedidos
From pedidos
```

A partir de esta sentencia calculamos la suma de los valores de todos los pedidos realizados y almacenamos ese valor en un campo virtual llamado suma_pedidos que podrá ser utilizado como cualquier otro campo por nuestras paginas dinámicas.

Por supuesto, todo lo visto hasta ahora puede ser aplicado en este tipo de funciones de modo que, por ejemplo, podemos establecer condiciones con la cláusula Where construyendo sentencias como esta:

```
Select Sum(cantidad) as suma_articulos
From pedidos Where id_articulo=6
```

Esto nos proporcionaría la cantidad de **ejemplares de un determinado libro que han sido vendidos**.

Otra propiedad interesante de estas funciones es que **permiten realizar operaciones con varios campos dentro de un mismo paréntesis**:

```
Select Avg(total/cantidad)
From pedidos
```

Esta sentencia da como resultado el **precio medio al que se están vendiendo los libros**. Este resultado no tiene por qué coincidir con el del **precio medio de los libros presentes en el inventario**, ya que, puede ser que la gente tenga tendencia a comprar los libros caros o los baratos:

```
Select Avg(precio) as precio_venta
From articulos
```

Una cláusula interesante en el uso de funciones es Group By. Esta cláusula nos permite agrupar registros a los cuales vamos a aplicar la función. Podemos por ejemplo calcular el **dinero gastado por cada cliente**:

```
Select id_cliente, Sum(total) as suma_pedidos
From pedidos
Group By id_cliente
```

O saber el **numero de pedidos que han realizado**:

```
Select id_cliente, Count(*) as numero_pedidos
From pedidos
Group By id_cliente
```