BASES DE DATOS

SQL - Consultas sencillas

SQL - Consultas sencillas

Capacidades

- Ejecuta las primeras consultas a una base de datos relacional.
- Selecciona información resumida y relevante para la resolución de problemas con necesidad de información.

Temas

- Sentencia SELECT
- Condiciones de selección

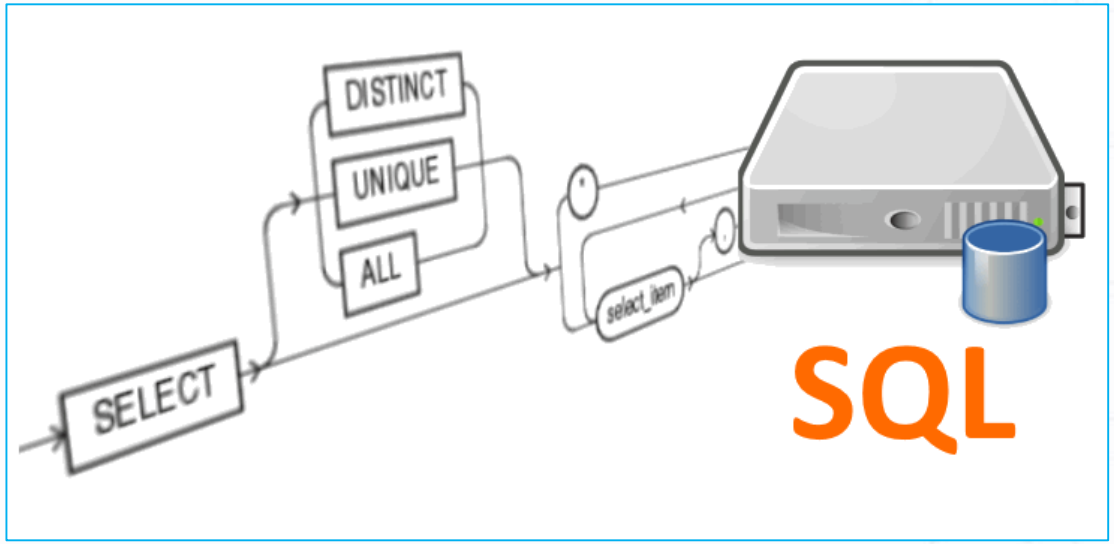
TEMA

Sentencia SELECT

¿En qué consisten las consultas en SQL?

En SQL las consultas nos permiten obtener información almacenada en una **base de datos**, podemos consultar los registros de una o varias tablas dependiendo de la información que necesitamos mostrar o dicho de otra manera podemos ver datos de una tabla. Una **consulta básica** o **sencilla** de SQL está enfocada en consultar datos en una sola tabla, y obtener filas y columnas en el orden y cantidad que nosotros queramos.

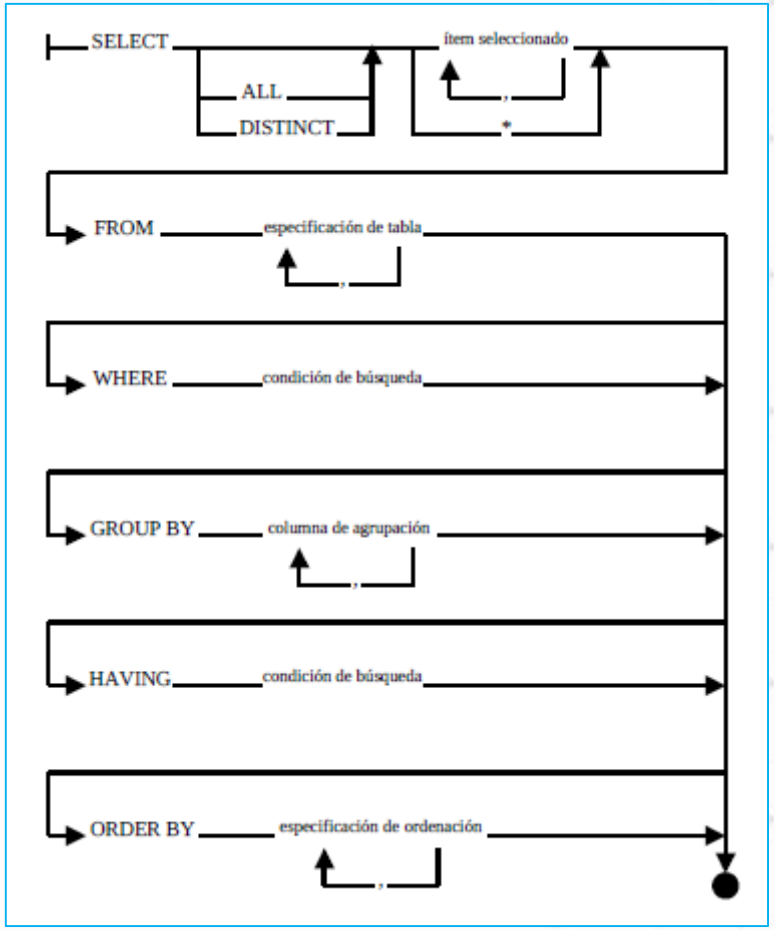
"El resultado de una consulta SQL es siempre una tabla de datos, semejante a las tablas de base de datos. Generalmente los resultados de la consulta formarán una tabla con varias columnas y filas".



La sentencia SELECT

La **sentencia SELECT** recupera datos de una base de datos y los devuelve en forma de resultados de la consulta. De acuerdo al gráfico adjunto, las **cláusulas *SELECT*** y ***FROM*** de la sentencia son necesarias, además, las cuatro cláusulas restantes son opcionales. La **cláusula *SELECT*** que inicia cada sentencia SELECT especifica los ítems de datos a recuperar por la consulta. Los **ítems** se especifican generalmente mediante una lista de selección separados por comas. Cada ítem de selección de la lista genera una única columna de resultados de consulta, en orden de izquierda a derecha. Un ítem puede ser:

- Un nombre de columna
- El valor *
- Una constante
- Una expresión SQL



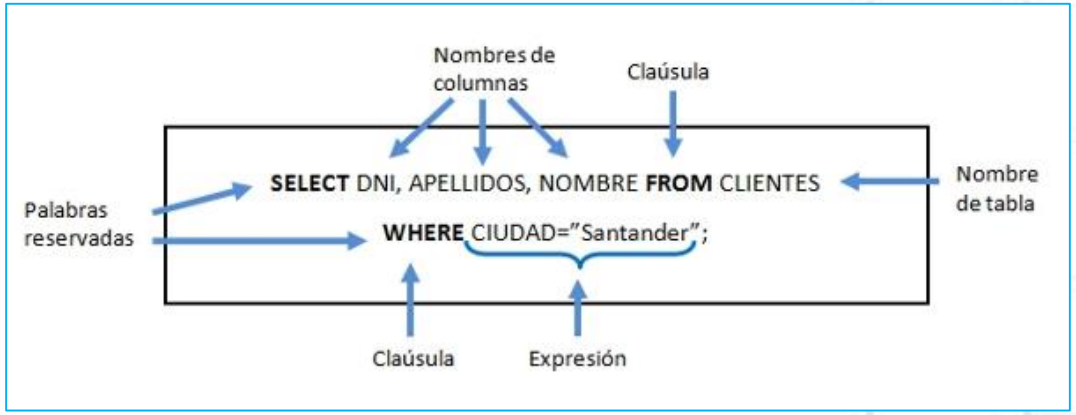
Sintaxis básica de la sentencia SELECT

La **sintaxis básica de la sentencia SELECT** es la que se muestra a continuación:

SELECT ALL | DISTINCT
{* | Expr_1 [AS e_alias_1] ,..., Expr_k [AS e_alias_k]}
FROM Table_1 [t_alias_1] ,..., Tabla_n [t_alias_n]

donde

- **Expr_i** : es el nombre del campo o la columna
- **e_alias_i** : es un alias para la expresión
- **Tabla_i** : es el nombre de la tabla de donde se extraerá la información
- **t_alias_i** : es un alias para la tabla



Cláusula FROM

Lista las tablas que contienen los datos a recuperar por la consulta. El formato de esta cláusula es:

SELECT * FROM Nombre_Tabla [Alias_Tabla] ,...

donde

- **Nombre_Tabla** : uno o más nombres de tabla en el directorio de trabajo
- **Alias_Tabla** : nombre que se usa para referirse a la tabla en el resto de la sentencia SELECT

```
SELECT  C.CategoryName, P.ProductName
 FROM  Products P,
     Categories C
```

En ocasiones es necesario recuperar información que no se encuentra contenida en la base de datos que actualmente está en uso. Un ejemplo de esto podría ser en que se muestra:

```
SELECT CategoryID, CategoryName
 FROM TSQL2019.dbo.Categories
```

donde **TSQL2019** es la base de datos que contiene la tabla **Categories**.

Cláusula ALL

Si no se incluye ningún campo o columnas se asume ALL. El motor de base de datos selecciona todos los registros que cumplen las condiciones de la instrucción SQL, pero **no es conveniente abusar de esta cláusula**, ya que obliga al motor de la base de datos, a analizar la estructura de la tabla para averiguar los campos que contiene.

```
SELECT ALL * FROM Products

/*Que sería lo mismo a:*/

SELECT * FROM Products
```

Cláusula DISTINCT

Omite los registros que contienen datos duplicados en los campos seleccionados. Para que los valores de cada campo listado en la sentencia SELECT se incluyan en la consulta, deben ser únicos. Con otras palabras, la **cláusula DISTINCT** devuelve aquellos registros cuyos campos indicados en la cláusula SELECT posean un contenido diferente. El resultado de una consulta que utiliza DISTINCT no es actualizable y no refleja los cambios subsiguientes realizados por otros usuarios.

```
SELECT DISTINCT Title
  FROM Employees
```

Función CONCAT

Sabemos que cuando queremos **concatenar** dos o más valores de distintos **campos** o **columnas** debemos hacerlo con el signo **+**. Por ejemplo:

SELECT Nombre + ' ' + Apellido AS [Nombre_completo] FROM Alumnos

Otra forma de hacer lo mismo es mediante la función llamada **CONCAT**, la cual une dos consultas dentro de una consulta, para que cuando se muestren, ya estén concatenados. Lo anterior mediante esta función sería

SELECT CONCAT(Nombre, ' ', Apellido) AS [Nombre_completo] FROM Alumnos

Función FORMAT

Retorna un valor con el formato previamente indicado. Se puede utilizar para definir el formato de una fecha y hora o para retornar un numero como carácter en un formato específico. Su formato es

FORMAT(Valor, Formato, [Cultura])

donde

- **Valor** : valor que se desea formatear
- **Formato** : define el formato del valor devuelto
- **Cultura** : valor opcional para obtener el resultado en nuestro idioma o lenguaje

TEMA

Condiciones de selección

¿Qué es la cláusula WHERE?

Esta cláusula dice a SQL, que incluya solo ciertas filas o registros de datos en los resultados de la consulta, es decir, que tienen que cumplir los registros que se desean ver. Contiene condiciones en la forma:

WHERE Expresión1 operador Expresion2

donde

- **Expresión1 y Expresión2** : nombres de campos, valores constantes o expresiones.
- **Operador** : operador relacional que une dos expresiones.

```
1  SELECT
2  SalesOrderDetailID,
3  OrderQty,
4  ProductID,
5  ModifiedDate
6  FROM Sales.SalesOrderDetail s
7  WHERE
8  ModifiedDate BETWEEN '2007/1/5' AND '2007/31/12' AND
9  ProductID = 809
```

100 %

Resultados | Mensajes

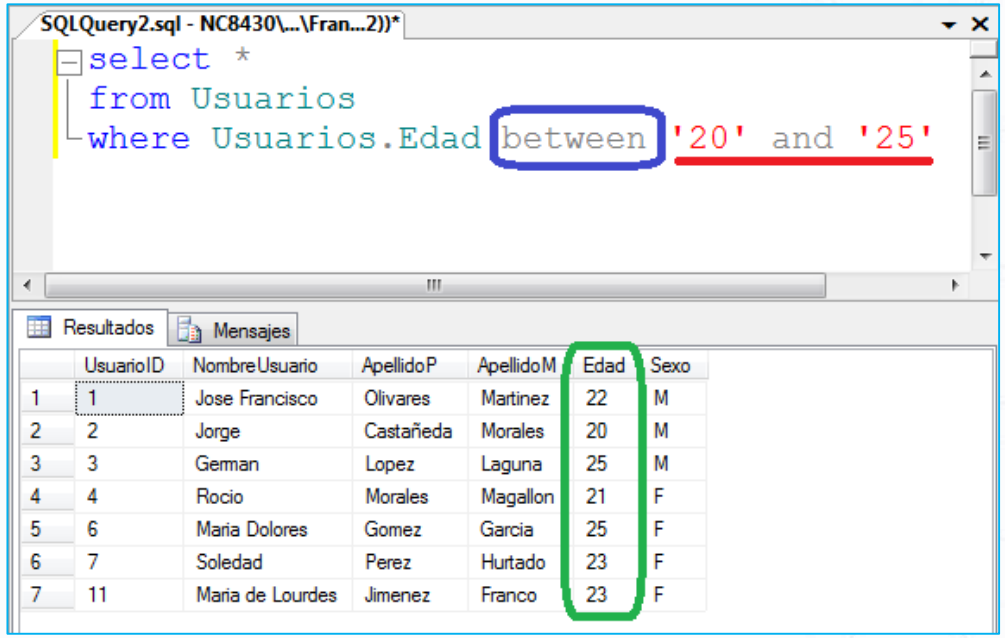
	SalesOrderDetailID	OrderQty	ProductID	ModifiedDate
1	30898	1	809	2007-05-01 00:00:00.000
2	30995	3	809	2007-05-01 00:00:00.000
3	31146	6	809	2007-05-01 00:00:00.000
4	31257	5	809	2007-05-01 00:00:00.000
5	31277	2	809	2007-05-01 00:00:00.000
6	31322	1	809	2007-05-01 00:00:00.000
7	31384	1	809	2007-05-01 00:00:00.000

WHERE con operadores de comparación

Para generar este tipo de sentencias de búsqueda en SQL, están disponibles los siguientes operadores de comparación:

- **>** : mayor
- **>=** mayor o igual
- **<** : menor
- **<=** : menor o igual
- **=** : igual
- **<>** o **!=** : distinto

También se puede usar **IS [NOT] NULL** para validar si el valor de una columna no es nulo, es decir, si contiene o no contiene algún registro y los operadores **BETWEEN**, **LIKE** e **IN**.



	UsuarioID	NombreUsuario	ApellidoP	ApellidoM	Edad	Sexo
1	1	Jose Francisco	Olivares	Martinez	22	M
2	2	Jorge	Castañeda	Morales	20	M
3	3	German	Lopez	Laguna	25	M
4	4	Rocio	Morales	Magallon	21	F
5	6	Maria Dolores	Gomez	Garcia	25	F
6	7	Soledad	Perez	Hurtado	23	F
7	11	Maria de Lourdes	Jimenez	Franco	23	F

WHERE con operadores lógicos booleanos

Para generar este tipo de sentencias de búsqueda en SQL, están disponibles los siguientes operadores lógicos booleanos:

- **AND** : "y lógico, evalúa dos condiciones y devuelve un valor de verdad, solo si ambas son ciertas.
- **OR** : "o" lógico, evalúa dos condiciones y devuelve un valor de verdadero, si alguna de las dos es cierta.
- **NOT** : negación lógica, devuelve el valor contrario de la expresión.

```
198  -- operador AND (Y)
199  SELECT id_producto, id_marca, id_categoria, nombre FROM productos
200  WHERE id_marca = 11 AND id_categoria = 618;
201
```

	id_producto	id_marca	id_categoria	nombre
1	9550	11	618	Tarjeta de Video
2	9551	11	618	Tarjeta de Video
3	9552	11	618	Tarjeta de Video
4	9553	11	618	Tarjeta de Video
5	9554	11	618	Tarjeta de Video
6	11482	11	618	Tarjeta de Video

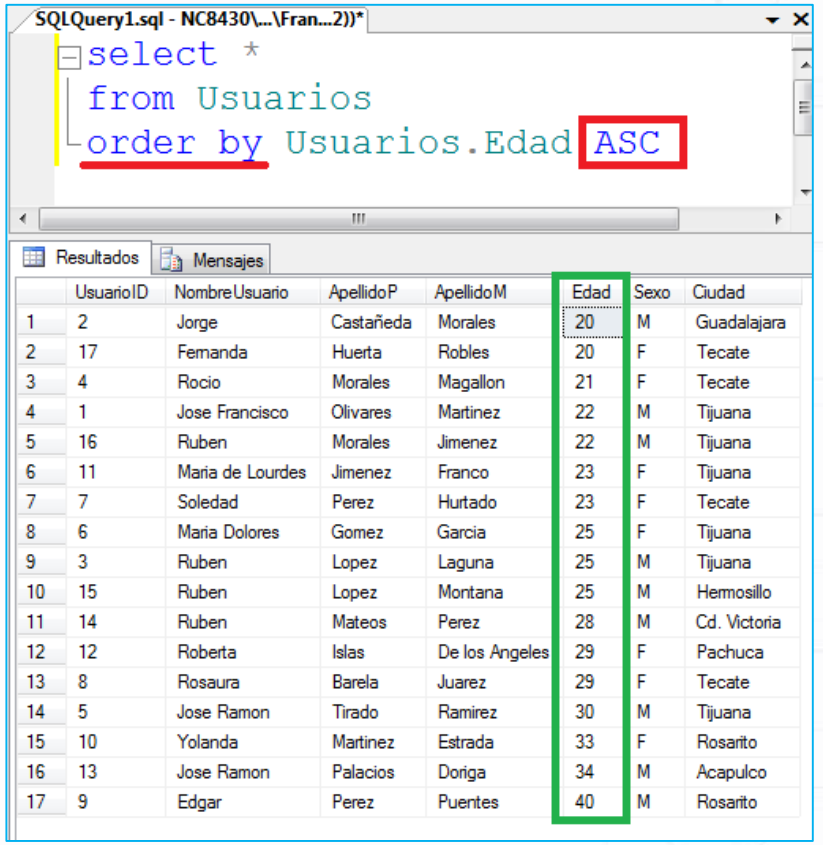
Cláusula ORDER BY

Ordena los resultados de la consulta con base a los datos de una o más columnas. Si se omite, los resultados saldrán ordenados por el primer campo que sea clave en el índice que se haya utilizado. Esta tiene la forma:

SELECT *
FROM {..., Nombre_Tabla_i ,...}
ORDER BY {..., Expresión_Orden_i [DESC | ASC] ,...}

donde

- **Nombre_Tabla_i** : uno o más nombres de tabla en el directorio de trabajo
- **Expresión_Orden_i [DESC | ASC]** : nombre de un campo, expresión o el número de posición que ocupa la expresión de columna en la cláusula SELECT



	UsuarioID	NombreUsuario	ApellidoP	ApellidoM	Edad	Sexo	Ciudad
1	2	Jorge	Castañeda	Morales	20	M	Guadalajara
2	17	Fernanda	Huerta	Robles	20	F	Tecate
3	4	Rocio	Morales	Magallon	21	F	Tecate
4	1	Jose Francisco	Olivares	Martinez	22	M	Tijuana
5	16	Ruben	Morales	Jimenez	22	M	Tijuana
6	11	Maria de Lourdes	Jimenez	Franco	23	F	Tijuana
7	7	Soledad	Perez	Hurtado	23	F	Tecate
8	6	Maria Dolores	Gomez	Garcia	25	F	Tijuana
9	3	Ruben	Lopez	Laguna	25	M	Tijuana
10	15	Ruben	Lopez	Montana	25	M	Hermosillo
11	14	Ruben	Mateos	Perez	28	M	Cd. Victoria
12	12	Roberta	Islas	De los Angeles	29	F	Pachuca
13	8	Rosaura	Barela	Juarez	29	F	Tecate
14	5	Jose Ramon	Tirado	Ramirez	30	M	Tijuana
15	10	Yolanda	Martinez	Estrada	33	F	Rosarito
16	13	Jose Ramon	Palacios	Doriga	34	M	Acapulco
17	9	Edgar	Perez	Puentes	40	M	Rosarito

¿Qué aprendimos?

Conclusiones

- Se definió la sentencia SELECT, la sintaxis que presenta junto con sus cláusulas y funciones más importantes.
- Se definió la cláusula WHERE la cual se utiliza para efectuar consultas mediante operadores de comparación y operadores lógicos booleanos. Además, se definió la cláusula ORDER BY para realizar ordenamientos.

```
77  -- Filtrar y ordenar los resultados por un campo o columna
78  SELECT id_producto, id_marca, nombre
79  FROM productos
80  WHERE id_marca = 363
81  ORDER BY nombre;
82
83
```

	id_producto	id_marca	nombre
1	13501	363	Adaptador
2	7457	363	Camara 2MP IR Turret Network
3	12529	363	Camara IP Bullet
4	7454	363	Camara 2MP HDCVI IR Dome
5	7458	363	Camara 2MP IR Turret Network