

Tema 1. Introducción (I)

¿Qué es el SQL ?

El **SQL** (Structured query language), **lenguaje de consulta estructurado**, es un lenguaje surgido de un proyecto de investigación de IBM para el acceso a bases de datos relacionales. Actualmente se ha convertido en un **estándar** de lenguaje de bases de datos, y la mayoría de los sistemas de bases de datos lo soportan, desde sistemas para ordenadores personales, hasta grandes ordenadores.

Por supuesto, a partir del estándar cada sistema ha desarrollado su propio SQL que puede variar de un sistema a otro, pero con cambios que no suponen ninguna complicación para alguien que conozca un SQL concreto, como el que vamos a ver aquí.

Como su nombre indica, el SQL nos **permite** realizar **consultas a la base de datos**. Pero el nombre se queda corto ya que SQL además realiza funciones de **definición, control y gestión de la base de datos**. Las sentencias SQL se clasifican según su finalidad dando origen a tres 'lenguajes' o mejor dicho sublenguajes:

• El **DDL** (Data Description Language), **lenguaje de definición** de datos, incluye órdenes para definir, modificar o borrar las tablas en las que se almacenan los datos y de las relaciones entre estas. (Es el que más varia de un sistema a otro)

• El **DCL** (Data Control Language), **lenguaje de control** de datos, contiene elementos útiles para trabajar en un entorno multiusuario, en el que es importante la protección de los datos, la seguridad de las tablas y el establecimiento de restricciones en el acceso, así como elementos para coordinar la compartición de datos por parte de usuarios concurrentes, asegurando que no interfieren unos con otros.

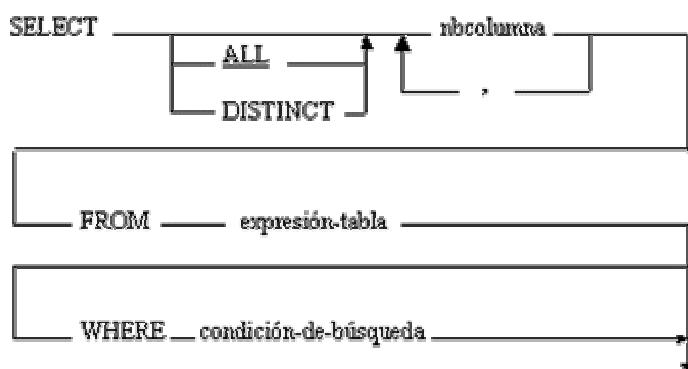
• El **DML** (Data Manipulation Language), **lenguaje de manipulación** de datos, nos permite recuperar los datos almacenados en la base de datos y también incluye órdenes para permitir al usuario actualizar la base de datos añadiendo nuevos datos, suprimiendo datos antiguos o modificando datos previamente almacenados.

Características del lenguaje

Una sentencia SQL es como una **frase** (escrita en **inglés**) con la que decimos **lo que queremos obtener y de donde obtenerlo**.

Todas las sentencias empiezan con un **verbo** (palabra reservada que indica la acción a realizar), seguido del resto de **cláusulas**, algunas **obligatorias** y otras **opcionales** que completan la frase. Todas las sentencias siguen una **sintaxis** para que se puedan ejecutar correctamente, para describir esa sintaxis utilizaremos un **diagrama sintáctico** como el que se muestra a continuación.

Cómo interpretar un diagrama sintáctico



Las palabras que aparecen en mayúsculas son palabras reservadas se tienen que poner tal cual y no se pueden utilizar para otro fin, por ejemplo, en el diagrama de la figura tenemos las palabras reservadas **SELECT, ALL, DISTINCT, FROM, WHERE**.

Las palabras en minúsculas son variables que el usuario deberá sustituir por un dato concreto. En el diagrama tenemos **nbcolumna, expresión-tabla y condición-de-búsqueda**.

Una sentencia válida se construye siguiendo la línea a través del diagrama hasta el punto que marca el final. Las líneas se siguen de **izquierda a derecha y de arriba abajo**. Cuando se quiere alterar el orden normal se indica con una **flecha**.

¿Cómo se interpretaría el diagrama sintáctico de la figura?

Hay que empezar por la palabra **SELECT**, después puedes poner **ALL** o bien **DISTINCT** o nada, a continuación un nombre de columna, o varios separados por comas, a continuación la palabra **FROM** y una expresión-tabla, y por último de forma opcional puedes incluir la cláusula **WHERE** con una condición-de-búsqueda.

Por ejemplo:

```

SELECT ALL col1,col2,col3 FROM mitabla
SELECT col1,col2,col3 FROM mitabla
SELECT DISTINCT col1 FROM mitabla
SELECT col1,col2 FROM mitabla WHERE col2 = 0

```

Todas estas sentencias se podrían escribir y no darían lugar a errores sintácticos.

Cuando una palabra opcional está **subrayada**, esto indica que ese es el **valor por defecto** (el valor que se asume si no se pone nada). En el ejemplo anterior las dos primeras sentencias son equivalentes (en el diagrama ALL aparece subrayada).

A lo largo del curso basaremos todos los ejemplos y ejercicios en las tablas que aparecen a continuación.

Nota: Estas tablas están orientadas a la didáctica, no a un diseño óptimo.

numemp	nombre	edad	oficina	titulo	contrato	jefe	cuota	ventas
101	Antonio Castro	45	12	Representante	20/10/1986	104	300.000	305.000
102	ARNOLDO AYALA	48	21	representante	10/12/1986	108	350.000	474.000
103	GUSTAVO FRANCO	29	12	representante	10/03/1987	104	275.000	286.000
104	OSCAR CARABALI	23	12	dir ventas	19/05/1987	106	200.000	143.000
105	VICENTE PATIÑO	37	13	representante	12/02/1988	104	350.000	368.000
106	RAFAEL NUÑEZ	52	11	dir general	14/06/1988		275.000	290.000
107	AURA ESTELLA E	49	22	representante	14/11/1988	108	300.000	186.000
108	CARLA MARIA TRUJILLO	62	21	dir ventas	12/10/1989	106	350.000	361.000
109	DEISY LILIANA LADINO	31	11	representante	12/10/1999	106	300.000	392.000
110	VICTOR SALAZAR	41		representante	13/01/1990	104	0	76.000

● Tabla **empleados** con los siguientes campos:

numemp: número del empleado

nombre: nombre y apellidos del empleado

edad: edad del empleado

oficina: número de la oficina donde trabaja el empleado, p.ej. Antonio trabaja en la oficina 12 de Alicante

titulo: el cargo que desempeña el empleado

contrato: fecha en que se contrató al empleado

jefe: número de su jefe inmediato, p.ej. El jefe de Antonio Viguer es José González. Observar que Luis Antonio no tiene jefe, es el director general.

cuota: cuota del empleado, sería el importe mínimo de ventas que debe alcanzar el empleado en el año

ventas: importe de ventas realizadas durante este año

oficina	ciudad	region	dir	objetivo	ventas
11	valencia	este	106	575.000	693.000
12	alicante	este	104	800.000	735.000
13	castellon	este	105	350.000	368.000
21	babajoz	oeste	108	725.000	836.000
22	a coruña	oeste	108	300.000	186.000
23	madrid	centro	108		
24	madrid	centro	108	250.000	150.000
26	pamplona	norte			
28	valencia	este		900.000	

Tabla **oficinas** con los siguientes campos:

oficina: código de la oficina

ciudad: ciudad donde está ubicada

región: región a la que pertenece

dir: director de la oficina (su número de empleado) **objetivo**: objetivo de ventas que debe alcanzar la oficina

ventas: ventas de la oficina

numclie	nombre	repclie	limitecredito
01	LUIS ANTONIO BENAVIDES	106	65.000
02	ARACELI MARTINEZ	101	65.000
03	JAIME ARREDONDO	105	50.000
04	MANUEL CABRERA	101	45.000
05	JUAN ESTEBAN SAMBRANO	102	65.000
06	JULIAN ANTONIO MARTINEZ	110	35.000
07	ARACELI MARTINEZ PULGARIN	109	55.000
08	ALRBETO SUAREZ	103	25.000
09	SAMUEL FERNANDO PERDOMO P	103	50.000
10	EDDIE LIVINGTON SEGURA C	104	50.000
11	LUIS AMARANTO PEREA	104	20.000
12	DENNY ALBERTO RIASCOS	102	20.000

13	FEDERI WISTON CABEZAS	101	20.000
14	CARLOTA SANCHE G	106	35.000
15	JHON BRANDON DIAZ	108	60.000
16	JUAN MANUEL OSPINA	109	25.000
17	JUAN CAMILO ESPINDOLA	102	50.000
18	VICENTE RIOS L	103	45.000
19	ERIKA CARDONA A	105	30.000
20	ANDREA PAOLA ALZATE	102	40.000
21	JUAN CARLOS PINTO	107	40.000

idfab	idproducto	descripcion	precio	existencias
aci	41001	arandela	58	277
aci	41002	bisagra	80	167
aci	41003	art t3	112	207
aci	41004	art t4	123	139
aci	4100x	junta	26	37
aci	4100y	extractor	2.888	25
aci	4100z	mont	2.625	28
bic	41003	manivela	652	3
bic	41089	rodamiento	225	78
bic	41672	plato	180	0
fea	112	cubo	148	115
fea	114	cubo	243	15
imm	773c	reostato	975	28
imm	775c	reostato2	1.425	5
imm	779c	reostato3	1.875	0
imm	887h	caja clavos	54	223
imm	887p	perno	25	24
imm	887x	manivela	475	32
qsa	xk47	red	355	38
qsa	xk48	red	134	203
qsa	xk48a	red	117	37
rei	2a44g	pas	350	14
rei	2a44i	bomba i	4.500	12
rei	2a44r	bomba r	4.500	12
rei	2a45c	junta	79	210

Tabla **productos** con los siguientes campos:

idfab: identificativo del fabricante del producto
idproducto : código que utiliza el fabricante para codificar el producto. Observar que aparecen varias líneas con el mismo idproducto (41003), por lo que la clave principal de la tabla deberá ser idfab+idproducto
descripcion: nombre del producto
precio: precio del producto
existencias: nº de unidades del producto que tenemos en almacén.

codigo	numpedido	fechapedido	clie	rep	fab	producto	cant	importe
1	110036	02/01/1997	06	110	aci	4100z	9	22.500
10	112992	15/04/1990	15	108	aci	41002	10	760
11	112993	10/03/1997	05	102	rei	2a45c	24	1.896
12	112997	04/04/1997	21	107	bic	41003	1	652
13	113003	05/02/1997	07	109	imm	779c	3	5.625
14	113007	01/01/1997	10	108	imm	773c	3	2.925
15	113012	05/05/1997	09	105	aci	41003	35	3.745
16	113013	06/08/1997	15	108	bic	41003	1	652
17	113024	04/07/1997	12	108	qsa	xk47	20	7.100
18	113027	05/02/1997	03	105	aci	41002	54	4.104
19	113034	05/11/1997	06	110	rei	2a45c	8	632
2	110036	02/04/1997	14	106	rei	2a44i	7	31.500
20	113042	01/01/1997	11	101	rei	2a44r	5	22.500
21	113045	02/07/1997	10	108	rei	2a44r	10	45.000
22	113048	02/02/1997	17	102	imm	779c	2	3.750
23	113049	04/04/1997	15	108	qsa	xk47	2	776
24	113051	06/07/1997	15	108	qsa	xk47	4	1.420
25	113055	01/04/2009	07	101	aci	4100x	6	150
26	113057	01/11/1997	09	103	aci	4100x	24	600

27	113058	04/07/1989	07	109	fea	112	10	1.480
28	113062	04/07/1997	21	107	bic	41003	10	2.430
29	113065	03/06/1997	05	102	qsa	xk47	6	2.130
3	112963	10/05/1997	03	105	aci	41004	28	3.276
30	113069	01/08/1997	08	107	imm	773c	22	31.350
4	112968	11/01/1990	02	101	aci	41004	34	3.978
5	112975	11/02/1997	09	103	rei	2a44g	6	2.100
6	112979	12/10/1989	12	108	aci	4100z	6	15.000
7	112983	10/05/1997	03	105	aci	41004	6	702
8	112987	01/01/1997	03	105	aci	4100y	11	27.500
9	112989	10/12/1997	01	106	fea	114	6	1.458

● Tabla pedidos:

codigo : nº secuencial que sirve de clave principal

numpedido: nº de pedido. Observar que un pedido puede tener varias líneas.

fechapedido : fecha del pedido

clie : cliente que efectúa el pedido

rep : representante que tramita el pedido

fab: fabricante del producto que se pide

producto : idproducto del producto que se pide.

cant : cantidad que se pide del producto

importe : importe de la línea de pedido

Tema 2. Las consultas simples (I)

Objetivo

Empezaremos por estudiar la sentencia **SELECT**, que permite **recuperar datos de** una o varias **tablas**. La sentencia SELECT es con mucho la más compleja y potente de las sentencias SQL.

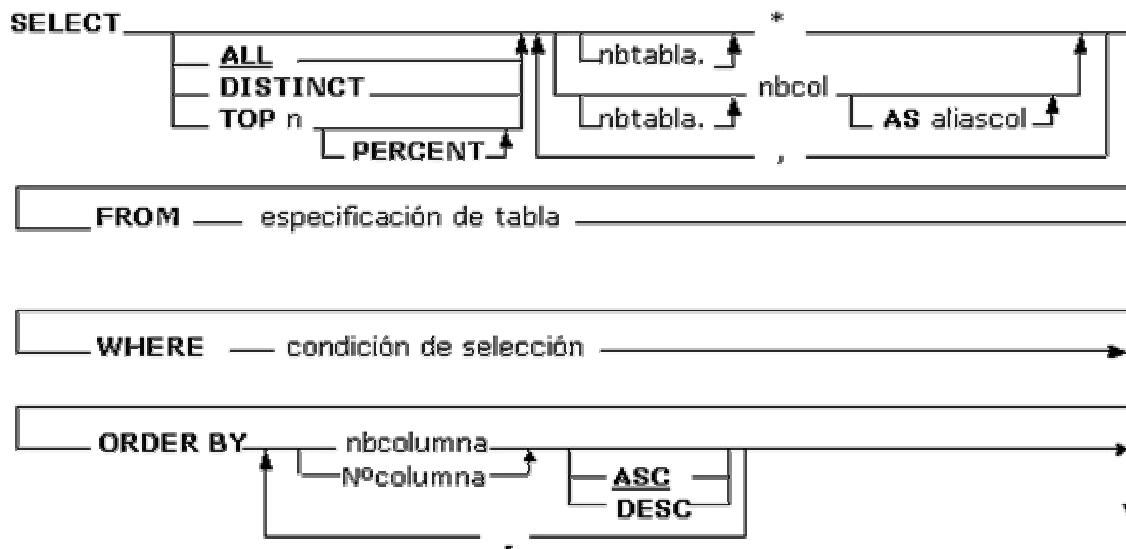
Empezaremos por ver las **consultas más simples, basadas en una sola tabla**.

Esta sentencia forma parte del DML (lenguaje de manipulación de datos), en este tema veremos cómo **seleccionar columnas** de una tabla, cómo **seleccionar filas** y cómo obtener las **filas ordenadas** por el criterio que queramos.

El resultado de la consulta es una **tabla lógica**, porque no se guarda en el disco sino que está en memoria y cada vez que ejecutamos la consulta se vuelve a calcular.

Cuando ejecutamos la consulta se visualiza el resultado en forma de tabla con columnas y filas, pues en la SELECT tenemos que indicar qué columnas queremos que tenga el resultado y qué filas queremos seleccionar de la tabla origen.

Sintaxis de la sentencia SELECT (consultas simples)

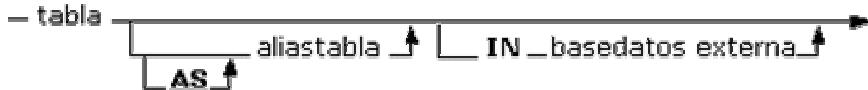


La tabla origen - FROM -

Con la cláusula **FROM** indicamos **en qué tabla** tiene que **buscar la información**. En este capítulo de consultas simples el resultado se obtiene de una única tabla. La sintaxis de la cláusula es:

FROM especificación de tabla

Una especificación de tabla puede ser el nombre de una consulta guardada (las que aparecen en la ventana de base de datos), o el nombre de una tabla que a su vez puede tener el siguiente formato:



Aliastabla es un nombre de alias, es como un **segundo nombre** que asignamos a la tabla, si en una consulta definimos un alias para la tabla, esta se deberá nombrar utilizando ese nombre y no su nombre real, además **ese nombre sólo es válido en la consulta** donde se define. El alias se suele emplear en consultas basadas en más de una tabla que veremos en el tema siguiente. La palabra AS que se puede poner delante del nombre de alias es opcional y es el valor por defecto por lo que no tienen ningún efecto.

Ejemplo: **SELECTFROM oficinas ofi** ; equivalente a **SELECTFROM oficinas AS ofi** esta sentencia me indica que se van a buscar los datos en la tabla oficinas que queda renombrada en esta consulta con ofi.

En una **SELECT** podemos utilizar tablas que no están definidas en la base de datos (siempre que tengamos los permisos adecuados claro), si la tabla no está en la base de datos activa, debemos indicar en qué base de datos se encuentra con la cláusula **IN**.

En la cláusula **IN** el nombre de la base de datos debe incluir el **camino completo**, la **extensión** (.mdb), y estar entre **comillas simples**.

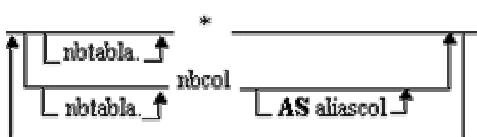
Supongamos que la tabla empleados estuviese en otra base de datos llamada otra en la carpeta c:\mis documentos\, habría que indicarlo así:

```
SELECT *
FROM empleados IN 'c:\mis documentos\otra.mdb'
```

Generalmente tenemos las tablas en la misma base de datos y no hay que utilizar la cláusula **IN**.

Selección de columnas

La lista de **columnas** que queremos que **aparezcan** en el **resultado** es lo que llamamos **lista de selección** y se especifica delante de la cláusula **FROM**.



Utilización del *

Se utiliza el asterisco * en la lista de selección para indicar '**todas las columnas de la tabla**'. Tiene dos **ventajas**:

Evitar nombrar las columnas una a una (es más corto).

Si añadimos una columna nueva en la tabla, esta nueva columna saldrá sin tener que modificar la consulta.

Se puede combinar el * con el nombre de una tabla (ej. oficinas.*), pero esto se utiliza más cuando el origen de la consulta son dos tablas.

```
SELECT * FROM oficinas
```

o bien

```
SELECT oficinas.* FROM oficinas
```

Lista todos los datos de las oficinas

columnas de la tabla origen

Las columnas se pueden especificar mediante su **nombre simple** (nbcoll) o su **nombre cualificado**

(ntablename.ncolumna, el nombre de la columna precedido del nombre de la tabla que contiene la columna y separados por un punto).

El nombre cualificado se puede emplear siempre que queramos y es obligatorio en algunos casos que veremos más adelante.

Cuando el **nombre** de la **columna** o de la tabla **contiene espacios en blanco**, hay que poner el nombre **entre corchetes** [] y además el número de espacios en blanco debe coincidir. Por ejemplo [codigo de cliente] no es lo mismo que [codigo de cliente] (el segundo lleva un espacio en blanco delante de código)

Ejemplos :

**SELECT nombre, oficina, contrato
FROM ofiventas**

Lista el nombre, oficina, y fecha de contrato de todos los empleados.

**SELECT idfab, idproducto, descripcion,
precio
FROM productos**

Lista una tarifa de productos

💡 Alias de columna.

Cuando se visualiza el resultado de la consulta, normalmente las columnas toman el nombre que tiene la columna en la tabla, si queremos cambiar ese nombre lo podemos hacer definiendo un alias de columna mediante la cláusula **AS** será el nombre que aparecerá como **título de la columna**.

Ejemplo:

**SELECT idfab AS fabricante,
idproducto, descripcion
FROM productos**

Como título de la primera columna aparecerá fabricante en vez de idfab

💡 Columnas calculadas.

Además de las columnas que provienen directamente de la tabla origen, una consulta SQL puede incluir **columnas calculadas** cuyos valores se calculan a partir de los valores de los datos almacenados.

Para solicitar una columna calculada, se especifica en la lista de selección una **expresión** en vez de un nombre de columna. La expresión puede contener sumas, restas, multiplicaciones y divisiones, concatenación & , paréntesis y también funciones predefinidas).

Ejemplos:

**SELECT ciudad, región, (ventas-
objetivo) AS superávit
FROM oficinas**

Lista la ciudad, región y el superávit de cada oficina.

**SELECT idfab, idproducto, descripcion,
(existencias * precio) AS valoracion
FROM productos**

De cada producto obtiene su fabricante, idproducto, su descripción y el valor del inventario

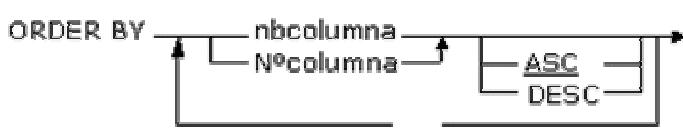
**SELECT nombre, MONTH(contrato),
YEAR(contrato)
FROM repventas**

Lista el nombre, mes y año del contrato de cada vendedor.
La función MONTH() devuelve el mes de una fecha
La función YEAR() devuelve el año de una fecha

**SELECT oficina, 'tiene
ventas de ', ventas
FROM oficinas**

Listar las ventas en cada oficina con el formato: 22 tiene ventas de 186,042.00 ptas

Ordenación de las filas - ORDER BY -



Para **ordenar** las **filas** del resultado de la consulta, tenemos la cláusula **ORDER BY**.

Con esta cláusula se altera el orden de visualización de las filas de la tabla pero en ningún caso se modifica el orden de las filas dentro de la tabla. La tabla no se modifica.

💡 Podemos indicar la columna por la que queremos ordenar utilizando su **nombre de columna** (nbcolumna) o utilizando su **número de orden** que ocupa en la lista de selección (Nºcolumna).

Ejemplo:

```
SELECT nombre, oficina,  
contrato  
FROM empleados  
ORDER BY oficina
```

es equivalente
a

```
SELECT nombre, oficina,  
contrato  
FROM empleados  
ORDER BY 2
```

💡 Por defecto el **orden** será **ascendente (ASC)** (de menor a mayor si el campo es numérico, por orden alfabético si el campo es de tipo texto, de anterior a posterior si el campo es de tipo fecha/hora, etc...)

Ejemplos:

```
SELECT nombre, numemp,  
oficinarep  
FROM empleados  
ORDER BY nombre
```

Obtiene un listado alfabético de los empleados.

```
SELECT nombre, numemp,  
contrato  
FROM empleados  
ORDER BY contrato
```

Obtiene un listado de los empleados por orden de antigüedad en la empresa (los de más antigüedad aparecen primero).

```
SELECT nombre, numemp, ventas  
FROM empleados  
ORDER BY ventas
```

Obtiene un listado de los empleados ordenados por volumen de ventas sacando los de menores ventas primero.

💡 Si queremos podemos alterar ese orden utilizando la cláusula **DESC** (DESCendente), en este caso el orden será el inverso al ASC.

Ejemplos:

```
SELECT nombre, numemp,  
contrato  
FROM empleados  
ORDER BY contrato DESC
```

Obtiene un listado de los empleados por orden de antigüedad en la empresa empezando por los más recientemente incorporados.

```
SELECT nombre, numemp, ventas  
FROM empleados  
ORDER BY ventas DESC
```

Obtiene un listado de los empleados ordenados por volumen de ventas sacando primero los de mayores ventas.

💡 También podemos **ordenar** por **varias columnas**, en este caso se indican las columnas separadas por comas.

Se ordenan las filas por la primera columna de ordenación, para un mismo valor de la primera columna, se ordenan por la segunda columna, y así sucesivamente.

La cláusula **DESC** o **ASC** se puede indicar para cada columna y así utilizar una ordenación distinta para cada columna. Por ejemplo ascendente por la primera columna y dentro de la primera columna, descendente por la segunda columna.

Ejemplos:

```
SELECT región, ciudad, ventas  
FROM oficinas  
ORDER BY región, ciudad
```

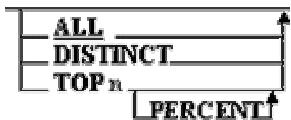
Muestra las ventas de cada oficina , ordenadas por orden alfabético de región y dentro de cada región por ciudad.

```
SELECT región, ciudad,  
(ventas - objetivo) AS superávit  
FROM oficinas  
ORDER BY región, 3 DESC
```

Lista las oficinas clasificadas por región y dentro de cada región por superávit de modo que las de mayor superávit aparezcan las primeras.

Selección de filas

A continuación veremos las cláusulas que nos permiten indicar **qué filas** queremos **visualizar**.



WHERE — condición-de-selección —

Las cláusulas DISTINCT / ALL

Al incluir la cláusula **DISTINCT** en la SELECT, se **eliminan** del resultado las **repeticiones** de filas. Si por el contrario queremos que aparezcan todas las filas **incluidas las duplicadas**, podemos incluir la cláusula **ALL** o nada, ya que ALL es el valor que SQL asume por defecto.

Por ejemplo queremos saber los códigos de los directores de oficina.

SELECT dir FROM oficinas
SELECT ALL dir FROM oficinas

Lista los códigos de los directores de las oficinas. El director 108 aparece en cuatro oficinas, por lo tanto aparecerá cuatro veces en el resultado de la consulta.

SELECT DISTINCT dir FROM oficinas

En este caso el valor 108 aparecerá una sola vez ya que le decimos que liste los distintos valores de directores.

La cláusula TOP

La cláusula **TOP** permite **sacar las n primeras filas** de la tabla origen. No elige entre valores iguales, si pido los 25 primeros valores pero el que hace 26 es el mismo valor que el 25, entonces devolverá 26 registros en vez de 25 (o los que sea). Siempre se guía por la columna de ordenación, la que aparece en la cláusula ORDER BY o en su defecto la clave principal de la tabla.

Por ejemplo queremos saber los dos empleados más antiguos de la empresa.

SELECT TOP 2 numemp, nombre
FROM empleado
ORDER BY contrato

Lista el código y nombre de los empleados ordenándolos por fecha de contrato, sacando únicamente los dos primeros (serán los dos más antiguos).

SELECT TOP 3 numemp, nombre
FROM empleado
ORDER BY contrato

En este caso tiene que sacar los tres primeros, pero si nos fijamos en las fechas de contrato tenemos 20/10/86, 10/12/86, 01/03/87, 01/03/87, la tercera fecha es igual que la cuarta, en este caso sacará estas cuatro filas en vez de tres, y sacaría todas las filas que tuviesen el mismo valor que la tercera fecha de contrato.

El número de filas que queremos visualizar se puede expresar con un número entero o como un **porcentaje** sobre el **número total de filas** que se recuperarían sin la cláusula TOP. En este último caso utilizaremos la cláusula **TOP n PERCENT** (porcentaje en inglés).

SELECT TOP 20 PERCENT nombre
FROM empleado
ORDER BY contrato

Lista el nombre de los empleados ordenándolos por fecha de contrato, sacando únicamente un 20% del total de empleados. Como tenemos 10 empleados, sacará los dos primeros, si tuviésemos 100 empleados sacaría los 20 primeros.

La cláusula WHERE

WHERE — condición-de-selección — La cláusula **WHERE** selecciona únicamente las **filas** que **cumplan la condición de selección** especificada.

En la consulta sólo aparecerán las filas para las cuales la condición es verdadera (TRUE), los valores nulos (NULL) no se incluyen por lo tanto en las filas del resultado. La **condición de selección** puede ser cualquier **condición válida** o **combinación de condiciones** utilizando los operadores **NOT** (no) **AND** (y) y **OR** (ó). En ACCESS2000 una cláusula WHERE puede contener

hasta 40 expresiones vinculadas por operadores lógicos AND y OR. Si quieres ver cómo funcionan los operadores lógicos

```
SELECT nombre  
FROM empleados  
WHERE oficina = 12
```

Lista el nombre de los empleados de la oficina 12.

```
SELECT nombre  
FROM empleados  
WHERE oficina = 12 AND edad > 30
```

Lista el nombre de los empleados de la oficina 12 que tengan más de 30 años. (oficina igual a 12 y edad mayor que 30)

Condiciones de selección

Las **condiciones de selección** son las condiciones que pueden aparecer en la cláusula WHERE.

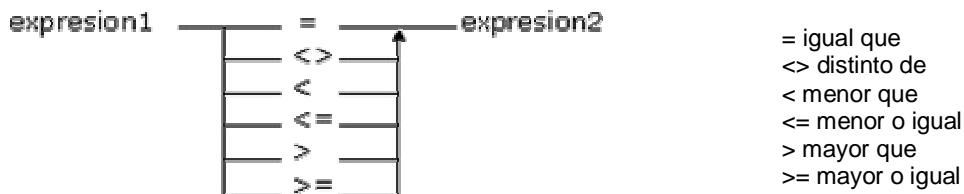
En SQL tenemos cinco condiciones básicas:

- el **test de comparación**
 - el **test de rango**
 - el **test de pertenencia a un conjunto**
 - el **test de valor nulo**
 - el **test de correspondencia con patrón**

El test de comparación.

Compara el valor de una expresión con el valor de otra.

La sintaxis es la siguiente:



```
SELECT numemp, nombre  
FROM empleados  
WHERE ventas > cuota
```

Lista los empleados cuyas ventas superan su cuota

```
SELECT numemp, nombre  
FROM empleados  
WHERE contrato < #01/01/1988#
```

Lista los empleados contratados antes del año 88 (cuya fecha de contrato sea anterior al 1 de enero de 1988).
¡¡Ojo!!, las fechas entre almohadillas # # deben estar con el formato mes,dia,año aunque tengamos definido otro formato para nuestras fechas.

```
SELECT numemp, nombre  
FROM empleados  
WHERE YEAR(contrato) < 1988
```

Este ejemplo obtiene lo mismo que el anterior pero utiliza la función year(). Obtiene los empleados cuyo año de la fecha de contrato sea menor que 1988.

```
SELECT oficina  
FROM oficinas  
WHERE ventas < objetivo * 0.8
```

Lista las oficinas cuyas ventas estén por debajo del 80% de su objetivo.
Hay que utilizar siempre el punto decimal aunque tengamos definida la coma como separador de decimales.

```
SELECT oficina  
FROM oficinas  
WHERE dir = 108
```

Lista las oficinas dirigidas por el empleado 108.

Test de rango (BETWEEN).

- **Test de rango (BETWEEN):**
Examina si el **valor** de la expresión está **comprendido entre** los **dos valores** definidos por exp1 y exp2. Tiene la siguiente sintaxis:

expression → **BETWEEN** → **exp1** → **AND** → **exp2** →
 └ **NOT** ┘

```
SELECT numemp, nombre  
FROM empleados  
WHERE ventas BETWEEN 100000 AND  
500000
```

Lista los empleados cuyas ventas estén comprendidas entre 100.000 y 500.00

```
SELECT numemp, nombre  
FROM empleados  
WHERE (ventas >= 100000) AND (ventas  
<= 500000)
```

Obtenemos lo mismo que en el ejemplo anterior. Los paréntesis son opcionales.

● Test de pertenencia a conjunto (IN)

Examina si el **valor** de la expresión es uno de los valores **incluidos en la lista de valores**. Tiene la siguiente sintaxis:

Expresión — IN — (valor) →
 ↑ , ↑

```
SELECT numemp, nombre, oficina  
FROM empleados  
WHERE oficina IN (12,14,16)
```

Lista los empleados de las oficinas 12, 14 y 16

```
SELECT numemp, nombre  
FROM empleados  
WHERE (oficina = 12) OR (oficina = 14)  
OR (oficina = 16)
```

Obtenemos lo mismo que en el ejemplo anterior. Los paréntesis son opcionales.

● Test de valor nulo (IS NULL)

Una condición de selección puede dar como resultado el valor verdadero TRUE, falso FALSE o nulo NULL.

Cuando una **columna** que interviene en una condición de selección **contiene el valor nulo**, el **resultado** de la condición no es verdadero ni falso, sino **nulo, sea cual sea el test** que se haya utilizado. Por eso si queremos listar las filas que tienen valor en una determinada columna, no podemos utilizar el test de comparación, la condición oficina = null devuelve el valor nulo sea cual sea el valor contenido en oficina. Si queremos preguntar si una columna contiene el valor nulo debemos utilizar un **test especial**, el test de valor nulo.

Tiene la siguiente sintaxis:

nbcolumna — IS — NULL — →
 ↑ NOT — →

Ejemplos:

```
SELECT oficina, ciudad  
FROM oficinas  
WHERE dir IS NULL
```

Lista las oficinas que no tienen director.

```
SELECT numemp, nombre  
FROM empleados  
WHERE oficina IS NOT NULL
```

Lista los empleados asignados a alguna oficina (los que tienen un valor en la columna oficina).

● Test de correspondencia con patrón (LIKE)

Se utiliza cuando queremos **utilizar caracteres comodines** para formar el valor con el comparar. Tiene la siguiente sintaxis:

nbcolumna — LIKE — patron — →
 ↑ NOT — →

Los comodines más usados son los siguientes:

? representa un carácter cualquiera

* representa cero o más caracteres

representa un dígito cualquiera (0-9)

Ejemplos:

```
SELECT numemp, nombre  
FROM empleados  
WHERE nombre LIKE 'Luis*' 
```

Lista los empleados cuyo nombre empiece por Luis (Luis seguido de cero o más caracteres).

```
SELECT numemp, nombre  
FROM empleados  
WHERE nombre LIKE '**Luis*' 
```

Lista los empleados cuyo nombre contiene Luis, en este caso también saldría los empleados José Luis (cero o más caracteres seguidos de LUIS y seguido de cero o más caracteres).

```
SELECT numemp, nombre  
FROM empleados  
WHERE nombre LIKE '??a*' 
```

Lista los empleados cuyo nombre contenga una a como tercera letra (dos caracteres, la letra a, y cero o más caracteres).

Ejercicios tema 2. Las consultas simples

Nota: Debes crear una consulta por cada ejercicio, no se pueden escribir varias sentencias SQL en una misma consulta.

Siquieres puedes guardar cada consulta con un nombre que permita identificarla por ejemplo: consulta_2_1 siendo 2 el número del tema y 1 el número del ejercicio dentro del tema.

Si la consulta contiene errores sintácticos no se podrá guardar.

Ahora puedes empezar a redactar las sentencias SQL para obtener lo que se pide en cada ejercicio.

La lista de selección

1 Obtener una lista de todos los productos indicando para cada uno su idfab, idproducto, descripción, precio y precio con I.V.A. incluido (es el precio anterior aumentado en un 16%).

2 De cada pedido queremos saber su número de pedido, fab, producto, cantidad, precio unitario e importe.

3 Listar de cada empleado su nombre, nº de días que lleva trabajando en la empresa y su año de nacimiento (suponiendo que este año ya ha cumplido años).

Ordenación de filas.

4 Obtener la lista de los clientes agrupados por código de representante asignado, visualizar todas la columnas de la tabla.

5 Obtener las oficinas ordenadas por orden alfabético de región y dentro de cada región por ciudad, si hay más de una oficina en la misma ciudad, aparecerá primero la que tenga el número de oficina mayor.

6 Obtener los pedidos ordenados por fecha de pedido.

Selección de filas.

7 Listar las cuatro líneas de pedido más caras (las de mayor importe).

8 Obtener las mismas columnas que en el ejercicio 2 pero sacando únicamente las 5 líneas de pedido de menor precio unitario.

9 Listar toda la información de los pedidos de marzo.

10 Listar los números de los empleados que tienen una oficina asignada.

11 Listar los números de las oficinas que no tienen director.

12 Listar los datos de las oficinas de las regiones del norte y del este (tienen que aparecer primero las del norte y después las del este).

13 Listar los empleados de nombre Julia.

14 Listar los productos cuyo idproducto acabe en x.