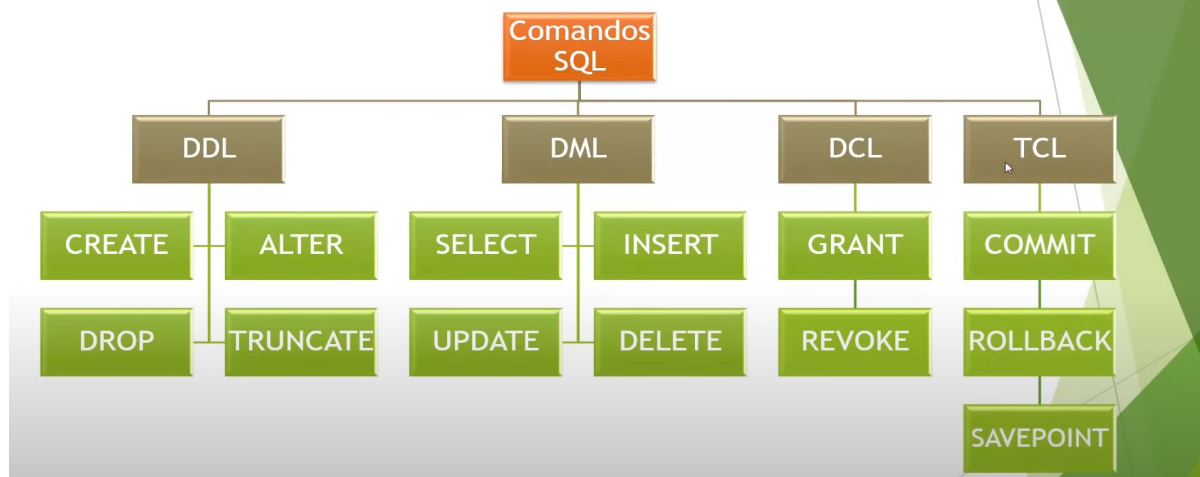


Temario

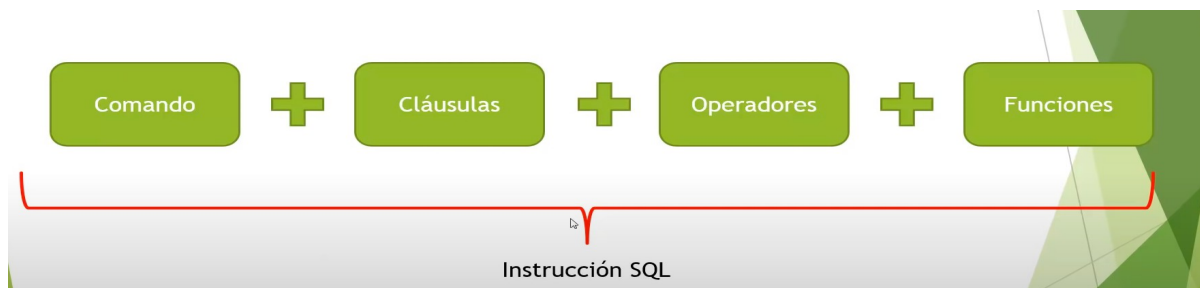
- ▶ Características del lenguaje SQL. Utilidad del lenguaje.
- ▶ Comandos SQL. Grupos de comandos.
- ▶ Consultas de selección.
 - ▶ Consultas multitable.
 - ▶ Consultas de agrupación o resumen.
 - ▶ Consultas de cálculo.
 - ▶ Subconsultas.
- ▶ Consultas de acción.
 - ▶ Creación de tabla.
 - ▶ Actualización.
 - ▶ Eliminación.
 - ▶ Datos anexados.
- ▶ Consultas de referencias cruzadas.
- ▶ Consultas de definición de datos.
 - ▶ Tipos de datos
 - ▶ Índices
 - ▶ Integridad referencial

Grupos de comandos



Cláusulas





- ▶ Cláusulas SQL
- ▶ Operadores
 - ▶ Comparación
 - ▶ Lógicos

Cláusulas

Cláusula	Descripción
FROM	Especifica la tabla de la que se quieren obtener los registros
WHERE	Especifica las condiciones o criterios de los registros seleccionados
GROUP BY	Para agrupar los registros seleccionados en función de un campo
HAVING	Especifica las condiciones o criterios que deben cumplir los grupos
ORDER BY	Ordena los registros seleccionados en función de un campo

Operadores de comparación

Operador	Significado
<	Menor que
>	Mayor que
=	Igual que
>=	Mayor o igual que
<=	Menor o igual que
<>	Distinto que
BETWEEN	Entre. Utilizado para especificar rangos de valores
LIKE	Cómo. Utilizado con caracteres comodín (? *)
In	En. Para especificar registros en un campo en concreto

Operadores lógicos

Operador	Significado
AND	Y lógico
OR	O lógico
NOT	Negación lógica

Orden de escritura



- Cláusula Order BY
- Consultas de agrupación o totales
 - Funciones de agregado

```
select count(CÓDIGOARTÍCULO),SECCIÓN, PAÍSDEORIGEN from productos where precio>50 GROUP BY PAÍSDEORIGEN,SECCIÓN HAVING SECCIÓN="deportes" ORDER BY PAÍSDEORIGEN DESC
```

☐ Mostrando todo | Número de filas: 25 | Filtrar filas:

+ Opciones

count(CÓDIGOARTÍCULO)	SECCIÓN	PAÍSDEORIGEN
5	DEPORTES	USA
1	DEPORTES	JAPÓN
1	DEPORTES	ESPAÑA

Ordenar por más de un campo:

```
SELECT * FROM PRODUCTOS WHERE SECCIÓN='DEPORTES' OR SECCIÓN='CERÁMICA' ORDER BY SECCIÓN, PAÍSDEORIGEN, PRECIO
```

Número de filas: 25

+ Opciones

CÓDIGOARTÍCULO	SECCIÓN	NOMBREARTÍCULO	PRECIO	FECHA	IMPORTADO	PAÍSDEORIGEN	FOTO
AR20	CERÁMICA	JUEGO DE TE	43.2728	2001-01-15	VERDADERO	CHINA	NULL
AR15	CERÁMICA	PLATO DECORATIVO	54.0911	2000-06-07	VERDADERO	CHINA	NULL
AR39	CERÁMICA	JARRA CHINA	127.7704	2002-09-02	VERDADERO	CHINA	NULL
AR11	CERÁMICA	TUBOS	168.4253	2000-02-04	VERDADERO	CHINA	NULL
AR33	CERÁMICA	MACETA	29.0434	2000-02-23	FALSO	ESPAÑA	NULL
AR21	CERÁMICA	CENICERO	19.7468	2001-07-02	VERDADERO	JAPÓN	NULL
AR41	DEPORTES	PALAS DE PING PONG	21.6000	2002-02-02	FALSO	ESPAÑA	NULL
AR28	DEPORTES	BALÓN FÚTBOL	43.9147	2002-07-04	FALSO	ESPAÑA	NULL
AR40	DEPORTES	BOTA ALPINISMO	144.0000	2002-05-05	FALSO	ESPAÑA	NULL
AR25	DEPORTES	BALÓN BALONCESTO	75.2731	2001-06-25	VERDADERO	JAPÓN	NULL
AR18	DEPORTES	PISTOLA OLIMPICA	46.7347	2001-02-02	VERDADERO	SUECIA	NULL
AR06	DEPORTES	MANCUERNAS	60.0000	2000-09-13	VERDADERO	USA	NULL
AR04	DEPORTES	RAQUETA TENIS	93.4694	2000-03-20	VERDADERO	USA	NULL
AR24	DEPORTES	BALÓN RUGBY	111.6440	2000-11-11	VERDADERO	USA	NULL
AR38	DEPORTES	CAÑA DE PESCA	270.0000	2000-02-14	VERDADERO	USA	NULL
AR32	DEPORTES	CRONÓMETRO	439.1764	2002-01-03	VERDADERO	USA	NULL

Funciones de agregado

Función	Descripción
AVG	Calcula el promedio de un campo
COUNT	Cuenta los registros de un campo
SUM	Suma los valores de un campo
MAX	Devuelve el máximo de un campo
MIN	Devuelve el mínimo de un campo

Importante: En las consultas de agrupación lo normal es sólo mostrar 2 campos: campo para calcular y campo para agrupar. Si metemos más, en MYSQL nos mostrará el primer artículo del campo por el que estemos agrupando tal como está en nuestra tabla de la BBDD.

```
select sección, sum(PRECIO) FROM productos GROUP by SECCIÓN order by sum(precio)
```

☐ Mostrar todo | Número de filas: 25 ▼ Filtrar filas:

+ Opciones

sección	sum(PRECIO)
OFICINA	39.76
FERRETERÍA	95.40
CERÁMICA	442.35
DEPORTES	1305.80
JUGUETERÍA	2516.72
CONFECCIÓN	501544.29

Con Alias:

```
select sección, sum(PRECIO) as Suma FROM productos GROUP by SECCIÓN order by Suma
```

☐ Mostrar todo | Número de filas: 25 ▼ Filtrar filas:

+ Opciones

sección	Suma ▲ 1
OFICINA	39.76
FERRETERÍA	95.40
CERÁMICA	442.35
DEPORTES	1305.80
JUGUETERÍA	2516.72
CONFECCIÓN	501544.29

Importante: En las consultas de agrupación para establecer criterios en vez de la cláusula where hay que utilizar la cláusula having

Ejemplo

```
SELECT SECCIÓN, AVG(PRECIO) AS MEDIA_ARTICULOS FROM PRODUCTOS GROUP BY SECCIÓN HAVING SECCIÓN='DEPORTES' OR SECCIÓN='CONFECCIÓN' ORDER BY MEDIA_ARTICULOS
```

☐ Perfilado

Número de filas: 25 ▼

+ Opciones

SECCIÓN	MEDIA_ARTICULOS ▲
DEPORTES	130.58123000
CONFECCION	55727.14372222

Otro ejemplo: La función count se utiliza sobre el campo clave porque no cuenta campos vacíos. Así nos aseguramos que todos los valores de esa columna/campo existen en todos los registros.

```
select paisdeorigen, sección, sum(PRECIO) as Suma, COUNT(CÓDIGOARTÍCULO) as N°Artículos FROM productos GROUP by PAISDEORIGEN,SECCIÓN order by paisdeorigen
```

☐ Mostrar todo | Número de filas: 25

+ Opciones

paisdeorigen	sección	Suma	N°Artículos
CHINA	CERÁMICA	393.56	4
CHINA	CONFECCIÓN	101.06	1
ESPAÑA	CERÁMICA	29.04	1
ESPAÑA	CONFECCIÓN	71.46	2
ESPAÑA	DEPORTES	209.51	3
ESPAÑA	FERRETERÍA	40.10	3
ESPAÑA	JUGUETERÍA	162.31	2
FRANCIA	CONFECCIÓN	30.20	1
FRANCIA	FERRETERÍA	9.06	1
ITALIA	CONFECCIÓN	500807.27	3
ITALIA	FERRETERÍA	6.74	1
JAPÓN	CERÁMICA	19.75	1
JAPÓN	DEPORTES	75.27	1
JAPÓN	JUGUETERÍA	1752.42	3
MARRUECOS	CONFECCIÓN	534.30	2
MARRUECOS	JUGUETERÍA	159.45	1
SUECIA	DEPORTES	46.73	1
TAIWÁN	FERRETERÍA	15.10	1
TURQUÍA	OFICINA	39.76	1
USA	DEPORTES	974.29	5
USA	FERRETERÍA	24.40	1
USA	JUGUETERÍA	442.54	1

Máximo y Mínimo:

```
select sección, max(PRECIO) as Precio_Máximo FROM productos GROUP by SECCIÓN HAVING SECCIÓN="confección"
```

☐ Mostrar todo | Número de filas: 25

+ Opciones

sección	Precio_Máximo
CONFECCIÓN	500000.00

En este caso también se podría hacer así:

```
select sección, max(PRECIO) as Precio_Máximo FROM productos where SECCIÓN="confección" GROUP by SECCIÓN
```

☐ Mostrar todo | Número de filas: 25

+ Opciones

sección	Precio_Máximo
CONFECCIÓN	500000.00

Para obtener el producto más caro de la sección de confección podríamos pensar que se haría así:

```
SELECT SECCIÓN, NOMBREARTÍCULO, MAX(PRECIO) AS PRECIO_MAS_ALTO FROM PRODUCTOS WHERE SECCIÓN='CONFECCIÓN' GROUP BY SECCIÓN
```

Número de filas: 25

+ Opciones

SECCIÓN	NOMBREARTÍCULO	PRECIO_MAS_ALTO
CONFECCIÓN	TRAJE CABALLERO	500000.0000

Pero de esta forma (hay más de 2 campos en el select de la consulta de agrupación) sólo obtenemos el primer nombre de artículo de los registros que hay en la tabla, pero no es el más caro porque estas consultas no funcionan de forma correcta cuando tenemos más de 2 campos (campo a agrupar más campo para calcular).

Esa consulta se haría de la siguiente manera, para obtener el artículo más caro de la sección de confección:

```
select nombrearticulo, sección, PRECIO as Precio_Máximo FROM productos where SECCIÓN="confección" order by precio desc LIMIT 1
```

+ Opciones

nombrearticulo	sección	Precio_Máximo
ABRIGO CABALLERO	CONFECCION	500000.00

► Consultas de cálculo

► Funciones frecuentes:

► Now() ←

► Datediff() ←

► Date_format() ←

► Concat()

ROUND---TRUNCATE

Las consultas de cálculo se realizan sobre registros individuales, no sobre grupos de registros, como sucedía con las consultas de agrupación:

Ejemplo1:

```
select NOMBREARTÍCULO, SECCIÓN, PRECIO, round(1.21*PRECIO,2) as PRECIO_IVA from productos
```

1 ▾

> >>

☐ Mostrar todo

Número de filas:

25 ▾

Filtrar filas:

Buscar en esta tabla

+ Opciones

NOMBREARTÍCULO	SECCIÓN	PRECIO	PRECIO_IVA
DESTORNILLADOR	FERRETERÍA	6.63	8.02
TRAJE CABALLERO	CONFECCION	284.58	344.34
COCHE TELEDIRIGIDO	JUGUETERÍA	159.45	192.93
RAQUETA TENIS	DEPORTES	93.47	113.10

Ejemplo2:

```
select NOMBREARTÍCULO, SECCIÓN, PRECIO, FECHA, DATE_FORMAT(now(), '%D-%M-%Y') as DÍA_HOY, datediff(now(), fecha) as DÍAS_HASTA_HOY from productos
```

1 ▾

> >>

☐ Mostrar todo

Número de filas:

25 ▾

Filtrar filas:

Buscar en esta tabla

+ Opciones

NOMBREARTÍCULO	SECCIÓN	PRECIO	FECHA	DÍA_HOY	DÍAS_HASTA_HOY
DESTORNILLADOR	FERRETERÍA	6.63	2000-10-22	23rd-June-2021	7549
TRAJE CABALLERO	CONFECCION	284.58	2002-03-11	23rd-June-2021	7044
COCHE TELEDIRIGIDO	JUGUETERÍA	159.45	2002-05-26	23rd-June-2021	6968
RAQUETA TENIS	DEPORTES	93.47	2000-03-20	23rd-June-2021	7765

En el ejemplo anterior usamos date_format para darle formato a la función now. La función now nos daría el día actual, pero también la hora actual. Con date_format le damos el formato que más nos convenga (se usa el símbolo de porcentaje antes de los días, meses o años). Datediff funciona de forma diferente para cada gestor de base de datos, en MySQL sólo admite dos parámetros, por lo que da siempre el número de días (en Access, por ejemplo el primer parámetro sería “D”, “M” o “YYYY”, para que la diferencia nos la de días, meses o años). Los parámetros que utiliza, para que el resultado sea un número positivo, son fecha más actual y luego fecha más antigua.

► Consultas Multitabla / Consultas de Unión

► Unión Externa:

- Union
- Union All
- Except
- Intersect
- Minus

► Unión interna

- Inner join
- Left Join
- Right Join

+

Unión:

Las consultas de unión sólo se pueden realizar entre tablas que tengan el mismo número de campos y estos ser compatibles (por ejemplo un campo numérico y un campo moneda son compatibles). Los campos pueden tener nombres diferentes. Los nombres de campo que se toman son los que había en la tabla uno.

Ejemplo1:

```
SELECT * FROM `productosnuevos` UNION SELECT * from productos
```

Ejemplo2: Se pueden escoger criterios en campos diferentes en cada una de las tablas.

Lo que si es que hay que indicar el mismo número de campos que se quieren visualizar en ambas tablas. Las consultas de unión se pueden hacer para más de 2 tablas operando de la misma manera.

```
SELECT * FROM `productosnuevos` WHERE SECCIÓN="DEPORTES DE RIESGO" UNION SELECT * FROM productos WHERE PRECIO>300
```

☐ Mostrar todo | Número de filas: 25 ▼ | Filtrar filas:

+ Opciones

CÓDIGOARTÍCULO	SECCIÓN	NOMBREARTÍCULO	PRECIO	FECHA	IMPORTADO	PAISDEORIGEN	FOTO
AR51	DEPORTES DE RIESGO	RAQUETA TENIS	1093.47	2000-03-20	VERDADERO	USA	NULL
AR52	DEPORTES DE RIESGO	MANCUERNAS	1060.00	2000-09-13	VERDADERO	USA	NULL
AR56	DEPORTES DE RIESGO	PISTOLA OLÍMPICA	1046.73	2001-02-02	VERDADERO	SUECIA	NULL
AR59	DEPORTES DE RIESGO	BALÓN RUGBY	1111.64	2000-11-11	VERDADERO	USA	NULL
AR60	DEPORTES DE RIESGO	BALON BALONCESTO	1075.27	2001-06-25	VERDADERO	JAPÓN	NULL
AR62	DEPORTES DE RIESGO	BALÓN FÚTBOL	1043.91	2002-07-04	FALSO	ESPAÑA	NULL
AR64	DEPORTES DE RIESGO	CRONOMETRO	1439.18	2002-01-03	VERDADERO	USA	NULL
AR66	DEPORTES DE RIESGO	CAÑA DE PESCA	1270.00	2000-02-14	VERDADERO	USA	NULL
AR67	DEPORTES DE RIESGO	BOTA ALPINISMO	1144.00	2002-05-05	FALSO	ESPAÑA	NULL
AR68	DEPORTES DE RIESGO	PALAS DE PING PONG	1021.60	2002-02-02	FALSO	ESPAÑA	NULL
AR10	JUGUETERÍA	CONSOLA VIDEO	442.54	2002-09-24	VERDADERO	USA	NULL
AR14	JUGUETERÍA	TREN ELÉCTRICO	1505.38	2001-07-03	VERDADERO	JAPÓN	NULL
AR23	CONFECCION	CAZADORA PIEL	522.69	2001-07-10	VERDADERO	ITALIA	NULL
AR27	CONFECCIÓN	ABRIGO CABALLERO	500000.00	2002-04-05	VERDADERO	ITALIA	NULL
AR29	CONFECCIÓN	ABRIGO SRA	360.07	2001-05-03	VERDADERO	MARRUECOS	NULL
AR32	DEPORTES	CRONÓMETRO	439.18	2002-01-03	VERDADERO	USA	NULL

Unión sólo muestra los registros una vez, mientras que UNION ALL muestra todos los registros repetidos tantas veces como aparezcan.

JOINS:

Ejemplo1:

SELECT * FROM pedidos as p join clientes as c on p.CÓDIGOCLIENTE=c.CÓDIGOCLIENTE where c.POBLOCIÓN="Madrid" ORDER BY 'p'.CÓDIGOCLIENTE ASC

Perfilando

Editar en línea

Editar

Explicar SQL

Crear código PHP

Actualizar

1 > >> ☐ Mostrar todo

Número de filas: 25

Filtrar filas:

Ordenar según la clave: Ninguna

+ Opciones

NÚMERO DE PEDIDO	CÓDIGOCLIENTE	FECHA DE PEDIDO	FORMA DE PAGO	DESCUENTO	ENVIADO	CÓDIGOCLIENTE	EMPRESA	DIRECCIÓN	POBLACIÓN	TÉLFONO	RESPONSABLE	HISTORIAL
1	CT01	2000-11-03	CONTADO	0	VERDADERO	CT01	BELTRÁN E HIJOS	LAS FUENTES 78	MADRID	914456435	ANGEL MARTÍNEZ	NULL
72	CT01	2002-08-18	CONTADO	0	VERDADERO	CT01	BELTRÁN E HIJOS	LAS FUENTES 78	MADRID	914456435	ANGEL MARTÍNEZ	NULL
73	CT01	2001-02-08	CONTADO	0	FALSO	CT01	BELTRÁN E HIJOS	LAS FUENTES 78	MADRID	914456435	ANGEL MARTÍNEZ	NULL
74	CT01	2002-09-17	APLAZADO	0	FALSO	CT01	BELTRÁN E HIJOS	LAS FUENTES 78	MADRID	914456435	ANGEL MARTÍNEZ	NULL
75	CT01	2002-09-30	TARJETA	0	FALSO	CT01	BELTRÁN E HIJOS	LAS FUENTES 78	MADRID	914456435	ANGEL MARTÍNEZ	NULL
76	CT01	2002-10-19	CONTADO	0	VERDADERO	CT01	BELTRÁN E HIJOS	LAS FUENTES 78	MADRID	914456435	ANGEL MARTÍNEZ	NULL
77	CT01	2000-10-28	CONTADO	0	FALSO	CT01	BELTRÁN E HIJOS	LAS FUENTES 78	MADRID	914456435	ANGEL MARTÍNEZ	NULL
8	CT01	2000-04-15	TARJETA	0	VERDADERO	CT01	BELTRÁN E HIJOS	LAS FUENTES 78	MADRID	914456435	ANGEL MARTÍNEZ	NULL
98	CT01	2001-12-27	CONTADO	0	VERDADERO	CT01	BELTRÁN E HIJOS	LAS FUENTES 78	MADRID	914456435	ANGEL MARTÍNEZ	NULL
40	CT04	2002-12-07	CONTADO	0	FALSO	CT04	EXPORTASA	VALLECAS 34	MADRID	913452378	ELVIRA GÓMEZ	NULL
85	CT04	2002-12-23	TARJETA	0	FALSO	CT04	EXPORTASA	VALLECAS 34	MADRID	913452378	ELVIRA GÓMEZ	NULL
11	CT04	2001-01-06	CONTADO	0	VERDADERO	CT04	EXPORTASA	VALLECAS 34	MADRID	913452378	ELVIRA GÓMEZ	NULL
22	CT07	2000-05-31	TARJETA	0	VERDADERO	CT07	LA CASA DEL JUGUETE	AMERICA 45	MADRID	912649987	ELIAS PEREZ	NULL
19	CT10	2002-05-22	CONTADO	0	VERDADERO	CT10	FERRETERÍA EL CLAVO	PASEO DE ÁLAMOS 78	MADRID	914354866	MANUEL MENÉNDEZ	NULL
32	CT14	2001-06-20	APLAZADO	0	FALSO	CT14	DEPORTES GARCÍA	GUZMÁN EL BUENO 45	MADRID	912799475	ANA JIMÉNEZ	NULL
3	CT23	2000-03-18	APLAZADO	0	FALSO	CT23	EL PALACIO DE LA MODA	ORTEGA Y GASSET 129	MADRID	927785235	LAURA CARRASCO	NULL
35	CT26	2001-06-30	CONTADO	0	FALSO	CT26	FERRETERÍA LA ESCOBA	ORENSE 7	MADRID	918459346	JOSÉ GARCÍA	NULL
34	CT26	2002-06-23	TARJETA	0	FALSO	CT26	FERRETERÍA LA ESCOBA	ORENSE 7	MADRID	918459346	JOSÉ GARCÍA	NULL
5005	CT30	2002-10-08	TARJETA	0	VERDADERO	CT30	BAZAR EL ARGENTINO	ATOCHA 55	MADRID	912495973	ADRIÁN ÁLVAREZ	NULL
105	CT30	2001-01-01	APLAZADO	0	FALSO	CT30	BAZAR EL ARGENTINO	ATOCHA 55	MADRID	912495973	ADRIÁN ÁLVAREZ	NULL
31	CT30	2000-08-06	TARJETA	0	VERDADERO	CT30	BAZAR EL ARGENTINO	ATOCHA 55	MADRID	912495973	ADRIÁN ÁLVAREZ	NULL
29	CT30	2001-02-04	TARJETA	0	FALSO	CT30	BAZAR EL ARGENTINO	ATOCHA 55	MADRID	912495973	ADRIÁN ÁLVAREZ	NULL
45	CT30	2002-07-22	TARJETA	0	FALSO	CT30	BAZAR EL ARGENTINO	ATOCHA 55	MADRID	912495973	ADRIÁN ÁLVAREZ	NULL
5050	CT30	2002-03-27	TARJETA	0	VERDADERO	CT30	BAZAR EL ARGENTINO	ATOCHA 55	MADRID	912495973	ADRIÁN ÁLVAREZ	NULL
44	CT34	2002-07-20	APLAZADO	0	FALSO	CT34	BAZAR LA FARAONA	CASTILLA Y LEÓN 34	MADRID	915483627	ANGEL SANTAMARIA	NULL

Ejemplo2: Sólo se muestran los clientes que han hecho pedidos (inner join)

SELECT DISTINCT P.CÓDIGOCLIENTE FROM pedidos as p join clientes as c on p.CÓDIGOCLIENTE=c.CÓDIGOCLIENTE

Mostrar todo

Número de filas: 25

Filtrar filas:

+ Opciones

CÓDIGOCLIENTE

CT01

CT02

CT04

CT06

CT07

CT09

CT10

CT12

CT13

CT14

CT16

CT18

CT20

CT21

CT23

CT24

CT25

CT26

CT28

CT30

CT31

CT34

Ejemplo 3: Se muestran también los clientes que no han hecho pedidos (right join en este caso).


```
SELECT DISTINCT C.CÓDIGOCLIENTE FROM pedidos as p RIGHT join clientes as c on p.CÓDIGOCLIENTE=c.CÓDIGOCLIENTE
```

1 > >> ☐ Mostrar todo | Número de filas: 25 Filtar filas:

+ Opciones

CÓDIGOCLIENTE

CT01

CT02

CT03

CT04

CT06

CT07

CT08

CT09

CT10

CT11

CT12

CT13

CT14

CT15

CT16

CT17

CT18

CT19

CT20

CT21

CT22

CT23

CT24

CT25

CT26

Se plantea lo siguiente: mostrar los clientes que son de Madrid y que no han realizado ningún pedido:

Solución mía:

```
SELECT C.CÓDIGOCLIENTE FROM CLIENTES AS C WHERE POBLACIÓN="MADRID" AND C.CÓDIGOCLIENTE NOT IN ( SELECT C.CÓDIGOCLIENTE FROM clientes AS C JOIN PEDIDOS AS P ON C.CÓDIGOCLIENTE=P.CÓDIGOCLIENTE WHERE C.POBLACIÓN="MADRID")
```

☐ Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código F

☐ Mostrar todo | Número de filas: 25 Filtar filas: Ordenar según la clave: Ninguna

+ Opciones

CÓDIGOCLIENTE

☐ Editar ☐ Copiar ☐ Borrar CT19

☐ Editar ☐ Copiar ☐ Borrar CT36

☐ Editar ☐ Copiar ☐ Borrar CT39

Solución él:

```
1 SELECT CLIENTES.CÓDIGOCLIENTE, POBLACIÓN, DIRECCIÓN, NÚMERODEPEDIDO, PEDIDOS.CÓDIGOCLIENTE, FORMADEPAGO FROM CLIENTES LEFT JOIN PEDIDOS ON CLIENTES.CÓDIGOCLIENTE=PEDIDOS.CÓDIGOCLIENTE WHERE POBLACIÓN='MADRID' AND PEDIDOS.CÓDIGOCLIENTE IS NULL
```

+ Opciones

CÓDIGOCLIENTE	POBLACIÓN	DIRECCIÓN	NÚMERODEPEDIDO	CÓDIGOCLIENTE	FORMADEPAGO
CT19	MADRID	FUENCARRAL 78	NULL	NULL	NULL
CT36	MADRID	ORENSE 89	NULL	NULL	NULL
CT39	MADRID	VALLECAS 45	NULL	NULL	NULL

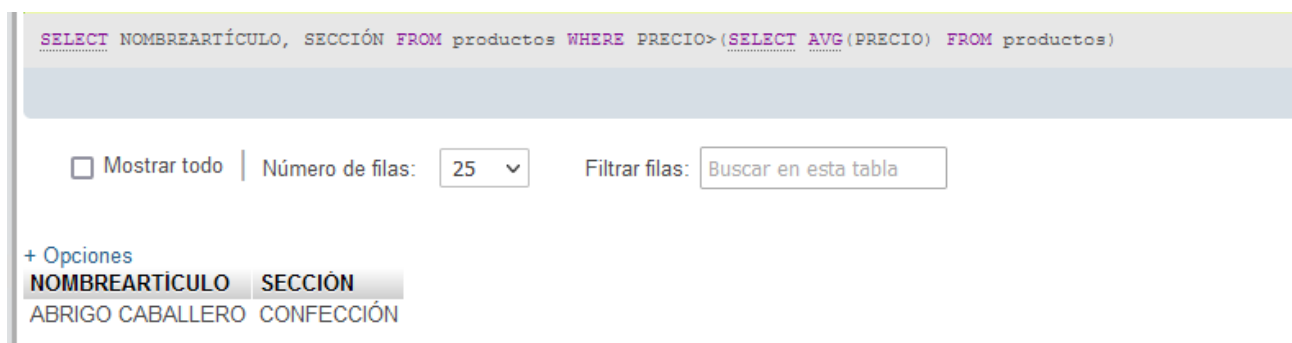
- ▶ Subconsultas ¿Qué son? Tipos
 - ▶ Subconsulta escalonada
 - ▶ Subconsulta de lista
 - ▶ Subconsulta correlacionada

Operadores lógicos y de comparación

- ▶ Lógicos
 - ▶ AND : todas las condiciones verdaderas
 - ▶ OR : una de las condiciones verdaderas
 - ▶ NOT : valor contrario de la expresión
- ▶ Comparación
 - ▶ LIKE : comparación de cadenas de caracteres
 - ▶ <>, <=, >=, <, > : mayor que..., menor que..., distinto que... etc
 - ▶ BETWEEN : intervalos
 - ▶ IN
 - ▶ ANY
 - ▶ ALL

Subconsulta Escalonada: el select interno devuelve una única columna con un único registro y ese resultado es el que se utiliza como criterio en la consulta padre.

Ejemplo1: Nombre del artículo y su sección de aquellos productos cuyo precio sea superior a la media.



The screenshot shows a database query interface. At the top, a SQL query is displayed: `SELECT NOMBREARTÍCULO, SECCIÓN FROM productos WHERE PRECIO > (SELECT AVG(PRECIO) FROM productos)`. Below the query, there are controls for displaying the results: a checkbox for "Mostrar todo", a "Número de filas:" dropdown set to "25", and a "Filtrar filas:" search box with the text "Buscar en esta tabla". At the bottom, there is a section for "+ Opciones" with two tabs: "NOMBREARTÍCULO" and "SECCIÓN". The "SECCIÓN" tab is active, showing a table with one row: "ABRIGO CABALLERO" under the "CONFECCIÓN" column.

Subconsultas de lista: En vez de devolvernos un único registro nos devuelve una lista de registros. Se utilizan los operadores in, any y all, en muchas ocasiones.

Ejemplo1: Queremos una consulta que nos devuelva los artículos cuyo precio es superior a todos los artículos de cerámica.

```
SELECT NOMBREARTÍCULO, SECCIÓN,PRECIO FROM productos WHERE PRECIO> ALL
(SELECT PRECIO FROM productos where SECCIÓN="cerámica" )
```

+ Opciones

NOMBREARTÍCULO	SECCIÓN	PRECIO
TRAJE CABALLERO	CONFECCIÓN	284.58
PANTALÓN SEÑORA	CONFECCIÓN	174.23
CONSOLA VIDEO	JUGUETERÍA	442.54
TREN ELÉCTRICO	JUGUETERÍA	1505.38
CAZADORA PIEL	CONFECCIÓN	522.69
ABRIGO CABALLERO	CONFECCIÓN	500000.00
ABRIGO SRA	CONFECCIÓN	360.07
CRONÓMETRO	DEPORTES	439.18
CAÑA DE PESCA	DEPORTES	270.00

Si pusiéramos un any lo que le estamos diciendo es que el precio sea mayor que cualquiera de los que pertenecen a la sección de cerámica. Nos daría un resultado con muchos más registros. Esta consulta también se podría hacer como consulta escalonada:

```
SELECT NOMBREARTÍCULO, SECCIÓN,PRECIO FROM productos WHERE  
PRECIO>( SELECT MAX(PRECIO) FROM productos where SECCIÓN="cerámica"  
)
```

► Predicados IN y NOT IN

Ejemplo: Nombre y precio de aquellos productos de los que se han pedido más de 20 unidades Usando JOIN:

```
SELECT NOMBREARTÍCULO, PRECIO FROM PRODUCTOS AS P JOIN  
PRODUCTOSPEDIDOS AS PP ON P.CÓDIGOARTÍCULO=PP.CÓDIGOARTÍCULO WHERE  
PP.UNIDADES>20.
```

Usando IN:

```
SELECT NOMBREARTÍCULO, PRECIO FROM PRODUCTOS WHERE CÓDIGOARTÍCULO  
IN (SELECT CÓDIGOARTÍCULO FROM PRODUCTOSPEDIDOS WHERE UNIDADES>20)
```

Nota: Es importante que en este segundo caso no necesitamos que las tablas estén relacionadas, en el caso de los JOINS sí, una tabla tiene que tener como clave foránea la clave primaria de la otra tabla.

Ejemplo2: Mostrar los clientes(EMRESA Y POBLACIÓN) que no han pagado con tarjeta o no han hecho pedidos.

Con NOT IN:

```
SELECT EMPRESA,POBLACIÓN FROM CLIENTES WHERE CÓDIGOCLIENTE NOT IN  
(SELECT CÓDIGOCLIENTE FROM PEDIDOS WHERE FORMADEPAGO="TARJETA")
```

`SELECT EMPRESA, POBLACIÓN FROM CLIENTES WHERE CÓDIGOCLIENTE NOT IN (SELECT CÓDIGOCLIENTE FROM PEDIDOS WHERE FORMADEPAGO="TARJETA")`

1 > >> | ☐ Mostrar todo | Número de filas: 25 | Filtrar filas: | Ordenar según la clave: Ninguna

EMPRESA	POBLACIÓN
LA MODERNA	OVIEDO
EL ESPAÑOLITO	BARCELONA
CONFECCIONES AMPARO	GUJÓN
JUGUETERÍA SUÁREZ	BARCELONA
ALMACÉN POPULAR	BILBAO
FERETERÍA EL CLAVO	MADRID
JUGUETES MARTÍNEZ	BARCELONA
FERNÁNDEZ SL	SANTANDER
CONFECCIONES ARTÍMEZ	A CORUÑA
DEPORTES GARCÍA	MADRID
EXCLUSIVAS FERNÁNDEZ	BARCELONA
DEPORTES MORÁN	LUGO
BAZAR FRANCISCO	ZAMORA
JUGUETES LA SONRISA	LEÓN
CONFECCIONES GALÁN	MADRID
LÍNEA JOVEN	SEVILLA
BAZAR EL BARAT	BARCELONA
EL PALACIO DE LA MODA	MADRID
DEPORTES EL MADRILEÑO	ZARAGOZA
JUGUETES EL BARATO	BARCELONA
CONFECCIONES HERMINIA	GUJÓN
LA TIENDA ELEGANTE	ZARAGOZA
DEPORTES NAUTICOS GARCÍA	ÁVILA
CONFECCIONES RUIZ	BARCELONA
BAZAR LA FARAONA	MADRID

CONSULTAS DE ACCIÓN:

► Consultas de acción:

- Actualización
- Creación de tabla
- Eliminación
- Datos anexados

► Comandos DML y DDL

- Create
- Update
- Delete
- Insert Into
- Select into

Consultas de Actualización:

Ejemplo1: Queremos incrementar en 10€ el precio de todos los artículos de la sección de Deportes:

`UPDATE PRODUCTOS SET PRECIO=(PRECIO+10) WHERE SECCIÓN="DEPORTES"`

Ejemplo2: Queremos cambiar el texto de la sección "DEPORTES" a "DEPORTIVOS".

`UPDATE productos SET SECCIÓN="DEPORTIVOS" WHERE SECCIÓN="DEPORTES"`

Consulta de creación de tabla a partir de otras:

Ejemplo1: Queremos crear una tabla con sólo los clientes de Madrid.

`CREATE TABLE CIENTESMADRID AS SELECT * FROM CLIENTES WHERE`

POBLACIÓN="MADRID"

(Nota: el "AS" no es obligatorio.)

En Access y SQL Server:

```
SELECT * INTO CLIENTESMADRID FROM CLIENTES WHERE POBLACIÓN="MADRID"
```

Create table es una instrucción más potente que la select into porque nos permite establecer que claves va a tener la nueva table, propiedades de algunos campos, etc. Select into se limita a introducir una serie de datos en una tabla nueva, sin determinar propiedades ni características de esa tabla nueva.

CONSULTAS DE ELIMINACIÓN:

Ejemplo1: Eliminar de la tabla de clientes los clientes de Madrid:

```
DELETE FROM CLIENTES WHERE POBLACIÓN="MADRID"
```

Ejemplo2: Borrar todos los productos de la sección de "DEPORTES" con un precio entre 50 y 100.

```
DELETE FROM PRODUCTOS WHERE SECCIÓN="DEPORTES" AND PRECIO BETWEEN 50 AND 100
```

Nota:

En las consultas de eliminación de tablas relacionadas hay que trabajar con consultas de predicado, los predicados son Distinct o Distinctrow

Ejemplo3: Queremos mostrar los clientes que si han realizado pedidos.

```
SELECT DISTINCT  
C.CÓDIGOCLIENTE,EMPRESA,DIRECCIÓN,POBLACIÓN,TELÉFONO,RESPONSABLE  
FROM CLIENTES AS C JOIN PEDIDOS AS P ON C.CÓDIGOCLIENTE=P.CÓDIGOCLIENTE  
ORDER BY C.CÓDIGOCLIENTE
```

DISTINCT: Para que no muestre la información repetida, para que los campos que se le indican no se repitan.

DISTINCTROW: Mira todo el registro, no repite registros.

Ejemplo 4: Queremos eliminar de la tabla de clientes aquellos que no han hecho pedidos.

```
DELETE FROM CLIENTES WHERE CÓDIGOCLIENTE NOT IN (SELECT  
CÓDIGOCLIENTE FROM PEDIDOS)
```

ACCESS con DISTINCTROW:

```
DELETE DISTINCTROW CLIENTE.*, P.CÓDIGOCLIENTE * FROM CLIENTES AS C LEFT  
JOIN PEDIDOS AS P ON C.CÓDIGOCLIENTE=P.CÓDIGOCLIENTE WHERE  
P.CÓDIGOCLIENTE IS NULL
```

CONSULTAS DE DATOS ANEXADOS:

Se utiliza el comando INSERT INTO.

Ejemplo 1: insert into clientesmadrid SELECT * from clientes where POBLACIÓN="barcelona"

Hemos introducido en esa tabla también los clientes de Barcelona.

Para deshacer el cambio: `DELETE FROM clientesmadrid WHERE POBLACIÓN="barcelona"`

Si no queremos introducir todos los campos.

`INSERT INTO clientesmadrid (CÓDIGOCLIENTE, EMPRESA, POBLACIÓN, TELÉFONO) SELECT CÓDIGOCLIENTE, EMPRESA, POBLACIÓN, TELÉFONO FROM clientes WHERE POBLACIÓN="barcelona"`

En SQL Server:

`SELECT * INTO CopiaEmpleado FROM Empleado`

`SELECT NSS,Nombre,Apel1,Apel2 INTO Emproy2 FROM Empleado WHERE NSS IN (SELECT NSS FROM EmpleadoProyecto WHERE NUMPROY=2)`

+ Opciones	CÓDIGOCLIENTE	EMPRESA	DIRECCIÓN	POBLACIÓN	TELÉFONO	RESPONSABLE	HISTORIAL
CT01		BELTRÁN E HIJOS	LAS FUENTES 78	MADRID	914456435	ANGEL MARTÍNEZ	NULL
CT04		EXPORTASA	VALLECAS 34	MADRID	913452378	ELVIRA GÓMEZ	NULL
CT07		LA CASA DEL JUGUETE	AMÉRICA 45	MADRID	912649987	ELÍAS PÉREZ	NULL
CT10		FERRETERÍA EL CLAVO	PASEO DE ÁLAMOS 78	MADRID	914354866	MANUEL MENÉNDEZ	NULL
CT14		DEPORTES GARCÍA	GUZMAN EL BUENO 45	MADRID	913299475	ANA JIMÉNEZ	NULL
CT19		CONFECCIONES GALÁN	FUENCARRAL 78	MADRID	913859234	JUAN GARCÍA	NULL
CT23		EL PALACIO DE LA MODA	ORTEGA Y GASSET 129	MADRID	927785235	LAURA CARRASCO	NULL
CT26		FERRETERÍA LA ESCOBA	ORENSE 7	MADRID	918459346	JOSE GARCÍA	NULL
CT30		BAZAR EL ARGENTINO	ATOCHA 55	MADRID	912495973	ADRIÁN ÁLVAREZ	NULL
CT34		BAZAR LA FARAONA	CASTILLA Y LEÓN 34	MADRID	915483627	ANGEL SANTAMARIA	NULL
CT36		JUGUETES EDUCATIVOS SANZ	ORENSE 89	MADRID	916872354	PEDRO IGLESIAS	NULL
CT39		FERRETERÍA LIMA	VALLECAS 45	MADRID	913532785	LUIS GARCÍA	NULL
CT03		EL ESPAÑOLITO	NULL	BARCELONA	934565343	NULL	NULL
CT08		JUGUETERÍA SUÁREZ	NULL	BARCELONA	933457866	NULL	NULL
CT11		JUGUETES MARTÍNEZ	NULL	BARCELONA	936628554	NULL	NULL
CT15		EXCLUSIVAS FERNÁNDEZ	NULL	BARCELONA	939558365	NULL	NULL
CT22		BAZAR EL BARAT	NULL	BARCELONA	936692866	NULL	NULL
CT27		JUGUETES EL BARATO	NULL	BARCELONA	933486984	NULL	NULL
CT33		CONFECCIONES RUIZ	NULL	BARCELONA	934587615	NULL	NULL
CT38		CONFECCIONES MONICA	NULL	BARCELONA	935681245	NULL	NULL

Hay que tener en cuenta que para que se produzca la anexión siempre hay que incluir en la anexión el campo clave y aquellos campos que sean requeridos.

REFERENCIAS CRUZADAS:



Con estas consultas podemos ver el resultado de las consultas en forma de tabla. (MYSQL no admite este tipo de consultas, sólo ACCESS):

Ejemplo1. Queremos mostrar estos 3 campos de la tabla productos:
NOMBREARTÍCULO,SECCIÓN,PRECIO

En la zona de totales es el campo que se va a operar (suma, media, máximo,etc). El campo precio será el que vaya en la zona de totales.

El NOMBREARTÍCULO iría en la zona de filas para que la tabla no sea tan alargada (son 39 artículos) y la sección en la zona superior (columnas) porque tenemos menos:

TRANSFORM SUM(PRECIO) AS TOTAL SELECT NOMBREARTÍCULO FROM PRODUCTOS GROUP BY NOMBREARTÍCULO PIVOT SECCIÓN

NOMBREARTÍCULO	CERÁMICA	CONFECCIÓN	DEPORTES	DEPORTIVOS	FERRETERÍA	JUGUETERÍA	OFICINA
ABRIGO CABALLERO		500.000,00 €					
ABRIGO SRA		360,07 €					
ALICATES					6,74 €		
BALÓN BALONCESTO				75,27 €			
BALÓN FÚTBOL				43,91 €			
BALÓN RUGBY				111,64 €			
BLUSA SRA.		101,06 €					
BOTA ALPINISMO				144,00 €			
CAMISA CABALLERO		67,13 €					
CAMISETA							
CAMISETA2							
CAMISETA3							
CAÑA DE PESCA				270,00 €			
CAZADORA PIEL		522,69 €					
CENICERO	19,75 €						
CINTURÓN DE PIEL		4,33 €					
COCHE TELEDIRIGIDO						159,45 €	
CONSOLA VIDEO						442,54 €	
CORREPASILLOS						103,34 €	
CRONÓMETRO				439,18 €			
DESTORNILLADOR					15,69 €		
FUERTE DE SOLDADOS						143,70 €	
JARRA CHINA	127,77 €						
JUEGO DE BROCAS					15,10 €		
JUEGO DE TE	43,27 €						
LIMA GRANDE					22,07 €		
LLAVE INGLESA					24,40 €		
MACETA	29,04 €						
MANCUERNAS				60,00 €			

Ejemplo2

TRANSFORM COUNT(CÓDIGOARTÍCULO) AS N_ARTICULOS SELECT NOMBREARTÍCULO FROM PRODUCTOS GROUP BY NOMBREARTÍCULO PIVOT SECCIÓN.

La anterior nos diría el número de artículos que aparecen repetidos.

Ejemplo3. Obtener el número de pedidos que ha hecho cada cliente con cada una de las formas de pago.(sería una consulta donde intervienen más de una tabla: clientes y pedidos)

SELECT EMPRESA,POBLACIÓN,FORMADEPAGO FROM CLIENTES INNER JOIN ON CLIENTES.CÓDIGOCLIENTE=PEDIDO.CÓDIGOCLIENTE

Esa consulta se guarda con un nombre, por ejemplo: “Previa”

Luego hacemos una consulta de referencias cruzadas, pero estaría basada en una consulta en vez de una tabla:

TRANSFORM COUNT(POBLACIÓN) AS TOTAL_FORMAPAGO SELECT EMPRESA FROM PREVIA GROUP BY EMPRESA PIVOT FORMADEPAGO.

Count(Población) lo incluimos porque es un campo que siempre está cubierto.

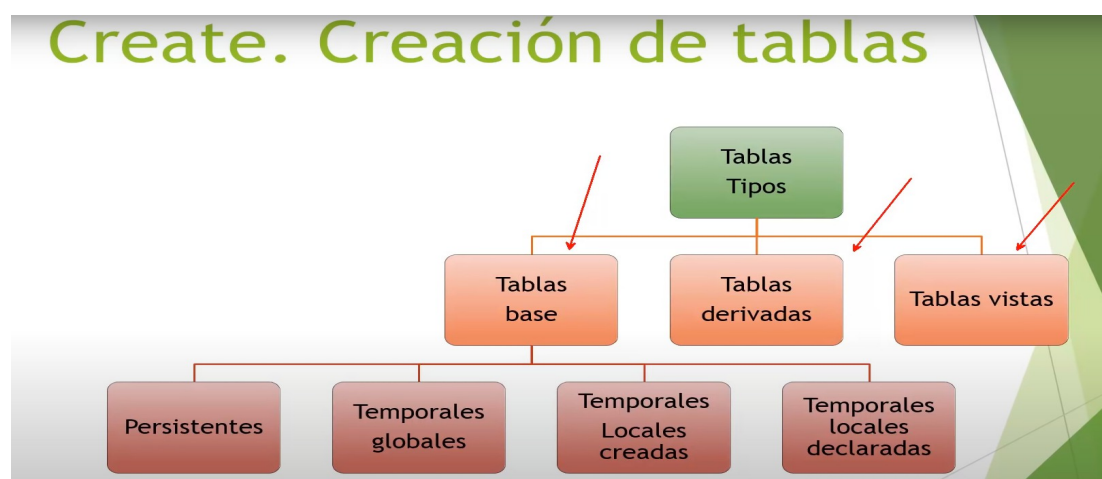
En la zona de filas se puede incluir más de un campo:

Ejemplo:La misma que el ejemplo 1, pero que en la zona de filas muestre 2 campos

TRANSFORM SUM(PRECIO) AS TOTAL SELECT NOMBREARTÍCULO,PAÍSDEORIGEN FROM PRODUCTOS GROUP BY NOMBREARTÍCULO,PAÍSDEORIGEN PIVOT SECCIÓN

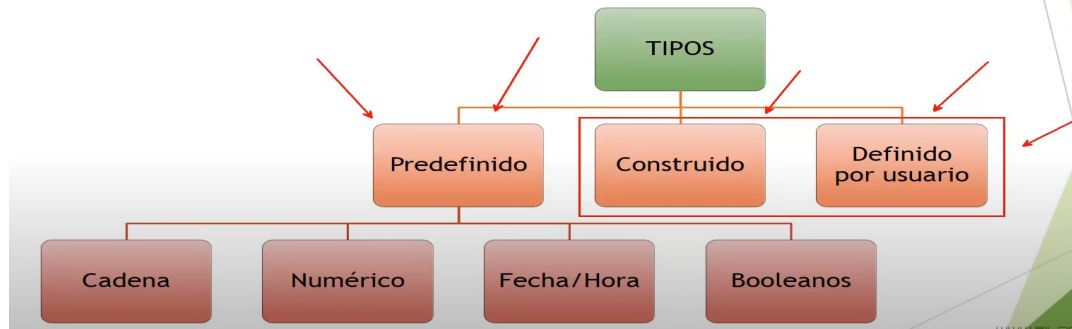
NOMBREARTÍCULO	PAÍSDEORIGEN	CERÁMICA	CONFECCIÓ	DEPORTES	DEPORTIVOS	FERRETERÍA	JUGUETERÍA	OFICINA
ABRIGO CABALLERO	ITALIA		500.000,00 €					
ABRIGO SRA	MARRUECOS		360,07 €					
ALICATES	ITALIA					6,74 €		
BALÓN BALONCESTO	JAPÓN				75,27 €			
BALÓN FÚTBOL	ESPAÑA				43,91 €			

- Comandos DDL. **D**ata **D**efinition **L**anguage. Definición de datos.



- `CREATE TABLE <NOMBRE DE LA TABLA> (CAMPO1 TIPO_DATO, CAMPO2 TIPO_DATO, CAMPO3 TIPO DATO...)`

TIPOS DE DATOS



TIPOS DE DATOS SQL

Tipo de dato	Sinónimos	Tamaño	Descripción
BINARY	VARBINARY BINARY VARYING BIT VARYING	1 byte por carácter	Se puede almacenar cualquier tipo de datos en un campo de este tipo. Los datos no se traducen (por ejemplo, a texto). La forma en que se introducen los datos en un campo binario indica cómo aparecerán al mostrarlos.
BIT	BOOLEAN LOGICAL LOGICAL1 YESNO	1 byte	Valores Sí y No, y campos que contienen solamente uno de dos valores.
TINYINT	INTEGER1 BYTE	1 byte	Número entero entre 0 y 255.
COUNTER	AUTOINCREMENT		Utilizado para campos contadores cuyo valor se incrementa automáticamente al crear un nuevo registro.
MONEY	CURRENCY	8 bytes	Un número entero comprendido entre -922.337.203.685.477,5808 y 922.337.203.685.477,5807.
DATETIME	DATE TIME	8 bytes	Un valor de fecha u hora entre los años 100 y 9999
UNIQUEIDENTIFIER	GUID	128 bits	Un número de identificación único utilizado con llamadas a procedimientos remotos.
DECIMAL	NUMERIC DEC	17 bytes	Un tipo de datos numérico exacto con valores comprendidos entre 1028 - 1 y - 1028 - 1. Puede definir la precisión (1 - 28) y la escala (0 - precisión definida). La precisión y la escala predeterminadas son 18 y 0, respectivamente.
REAL	SINGLE FLOAT4 IEEE SINGLE	4 bytes	Un valor de coma flotante de precisión simple con un intervalo comprendido entre -3,402823E38 y -1,401298E-45 para valores negativos, y desde 1,401298E-45 a 3,402823E38 para valores positivos, y 0.
FLOAT	DOUBLE FLOAT8	8 bytes	Un valor de coma flotante de precisión doble con un intervalo comprendido entre -1,79769313486232E308 y -4,94065645841247E-324 para valores negativos, y desde 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos, y 0.

	IEEEDOUBLE NUMBER		
SMALLINT	SHORT INTEGER2	2 bytes	Entero corto entre - 32.768 y 32.767.
INTEGER	LONG INT INTEGER4	4 bytes	Entero largo entre - 2.147.483.648 y 2.147.483.647.
IMAGE	LONGBINARY GENERAL OLEOBJECT	Lo que se requiera	De cero hasta un máximo de 2.14 gigabytes. Se utiliza para objetos OLE.
TEXT	LONGTEXT LONGCHAR MEMO NOTE NTEXT	2 bytes por carácter. (Consulte las notas).	De cero hasta un máximo de 2.14 gigabytes.
CHAR	TEXT(n) ALPHANUMERIC CHARACTER STRING VARCHAR CHARACTER VARYING NCHAR NATIONAL CHARACTER NATIONAL CHAR NATIONAL CHARACTER VARYING NATIONAL CHAR VARYING	2 bytes por carácter. (Consulte las notas).	De cero a 255 caracteres.

Ejemplo 1:

CREATE TABLE PRUEBA (NOMBRE VARCHAR(20))

CREATE TABLE CLLENTESMADRID AS SELECT * FROM CLIENTES WHERE POBLACIÓN="MADRID"

En Acces:

CREATE TABLE PRUEBA(NOMBRE TEXT(20))

Eliminar la tabla que creamos antes en MYSQL:

DROP TABLE PRUEBA

Ejemplo2 con campo con autoincremento: Creamos una tabla con más campos

CREATE TABLE PRUEBA (ID_ALUMNO INT AUTO_INCREMENT, NOMBRE VARCHAR(20), APELLIDO VARCHAR(20), EDAD TINYINT, FECHA_NACIMIENTO DATE, CARNET BOOLEAN, PRIMARY KEY (ID_ALUMNO))

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/> 1	ID_ALUMNO	int(11)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Primar
<input type="checkbox"/> 2	NOMBRE	varchar(20)	utf8mb4_general_ci		Si	NULL			Cambiar Eliminar Primar
<input type="checkbox"/> 3	APELLIDO	varchar(20)	utf8mb4_general_ci		Si	NULL			Cambiar Eliminar Primar
<input type="checkbox"/> 4	EDAD	tinyint(4)			Si	NULL			Cambiar Eliminar Primar
<input type="checkbox"/> 5	FECHA_NACIMIENTO	date			Si	NULL			Cambiar Eliminar Primar
<input type="checkbox"/> 6	CARNET	tinyint(1)			Si	NULL			Cambiar Eliminar Primar

En Access con campo autonumérico:

CREATE TABLE PRUEBA (ID_ALUMNO COUNTER, NOMBRE TEXT(20), APELLIDO TEXT(20), EDAD BYTE, FECHA_NACIMIENTO, CARNET BIT)

Campo autonumérico en Access (counter) y en MYSQL (int auto_increment y poniendo al final de todos los campos que ese campo será clave primaria)

- ▶ Agregar, modificar y eliminar campos en una tabla
- ▶ Valores por defecto de los campos de una tabla

Alter y Drop:



Añadimos un campo nuevo a la tabla clientes para indicar la fecha en la que les damos de baja:

[ALTER TABLE](#) clientes ADD column FECHA_BAJA [DATE](#)

En Access sería exactamente igual

Eliminar un campo de una tabla: Eliminamos el campo Población de la tabla Prueba

[ALTER TABLE](#) prueba DROP COLUMN POBLACIÓN

Modificar características de un campo, por ejemplo el tipo de datos:

Access:

[ALTER TABLE](#) clientes ALTER column FECHA_BAJA [varchar](#)(20)

MYSQL:

[ALTER TABLE](#) clientes MODIFY column FECHA_BAJA [varchar](#)(20)

Establecer el valor por defecto en un campo de una tabla. En MYSQL (Access no permite establecer este valor desde SQL).

En este caso establecemos LUGAR_NACIMIENTO como 'DESCONOCIDO', pasaría de NULL a ese valor

ALTER TABLE prueba ALTER COLUMN LUGAR_NACIMIENTO SET DEFAULT 'DESCONOCIDO'

Si queremos eliminar ese valor predeterminado usamos el comando DROP:

ALTER TABLE prueba ALTER COLUMN LUGAR_NACIMIENTO DROP DEFAULT

ÍNDICES: Nos permiten realizar búsquedas en una tabla de una base de datos con mayor rapidez. Se notará si la tabla tiene miles y miles de registros.

- Creación de índices:
 - Índices de clave primaria
 - Índices ordinarios
 - Índices únicos
 - Índices compuestos

Grupos de comandos



Índices de clave primaria (todo igual para Access y Mysql):

1º Creamos una tabla:

CREATE TABLE EJEMPLO (DNI VARCHAR(10), NOMBRE VARCHAR(20), APELLIDO VARCHAR(30), EDAD TINYINT, PRIMARY KEY (DNI))

Borramos la clave: ALTER TABLE ejemplo DROP PRIMARY KEY;

Agregando la clave a posteriori:

ALTER TABLE EJEMPLO ADD PRIMARY KEY (DNI)

Clave primaria formada por más de un campo (multicampo):

ALTER TABLE EJEMPLO ADD PRIMARY KEY (NOMBRE, APELLIDO)

Índices ordinarios: El comportamiento es como un campo normal, pero nos va a permitir realizar búsquedas de forma más rápida.

[ALTER TABLE](#) EJEMPLO ADD KEY (APELLIDO)

O

CREATE INDEX MIINDICE ON EJEMPLO (APELLIDO)

Diferencia entre uno y otros (sólo en el nombre que se le dio al índice):

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1	DNI	varchar(10)	utf8mb4_general_ci	No	Ninguna			Cambiar Eliminar Primaria Único Índice
<input type="checkbox"/>	2	NOMBRE	varchar(20)	utf8mb4_general_ci	No	Ninguna			Cambiar Eliminar Primaria Único Índice
<input type="checkbox"/>	3	APELLIDO	varchar(30)	utf8mb4_general_ci	Sí	NULL			Cambiar Eliminar Primaria Único Índice
<input type="checkbox"/>	4	EDAD	tinyint(4)		Sí	NULL			Cambiar Eliminar Primaria Único Índice

Seleccionar todo Para los elementos que están marcados: Examinar Cambiar Eliminar Primaria Único Índice Agregar a columna

Imprimir Planteamiento de la estructura de tabla Hacer seguimiento a la tabla Mover columnas Mejorar la estructura de tabla

Agregar 1 columna(s) después de EDAD Continuar

Índices

Acción	Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
Editar Eliminar	APELLIDO	BTREE	No	No	APELLIDO	0	A	Sí	
Editar Eliminar	MIINDICE	BTREE	No	No	NOMBRE	0	A	No	

Índices únicos:

Borramos los índices anteriores para crear un índice único con el campo apellido:

ALTER TABLE ejemplo DROP INDEX APELLIDO;

ALTER TABLE ejemplo DROP INDEX MIINDICE;

Creamos el índice único:

CREATE UNIQUE INDEX MIINDICE ON EJEMPLO (APELLIDO)

O ASÍ:

[ALTER TABLE](#) EJEMPLO ADD UNIQUE KEY (APELLIDO)

ÍNDICES COMPUESTOS: No permiten que estén duplicados, pero si que pueden estar vacíos.

Es decir en el caso siguiente, no puede haber personas que compartan nombre y apellido, pero al no ser de clave primaria no es obligatorio cubrir esos campos.

CREATE INDEX MIINDICE ON EJEMPLO (NOMBRE, APELLIDO)

O ASÍ:

[ALTER TABLE](#) EJEMPLO ADD KEY (NOMBRE, APELLIDO)

ELIMINACIÓN DE ÍNDICES:

Eliminación clave primaria: [ALTER TABLE](#) ejemplo DROP PRIMARY KEY

ACCESS:: ALTER TABLE EMJEMPLO DROP CONSTRAINT + (nombre que el gestor de BBDD le dio al índice, por ejemplo Index_2E945B32_F3AF_4D42)

Ejemplo para eliminar otro tipo de índice:

Access: DROP INDEX MIINDICE ON EJEMPLO

o MySQL: [ALTER TABLE](#) ejemplo [DROP INDEX](#) NOMBRE

TRIGGERS: Access no tiene Triggers, pero a partir de la versión de 2010 tiene unas macros nuevas que suplen esa carencia.

► Qué son los triggers

- Tipos
- Utilidad
- Creación

- TRIGGER de actualización
- Diferencias entre AFTER Y BEFORE
- Uso de OLD Y NEW

- TRIGGER de eliminación.
- Eliminación de triggers.
- Modificación de triggers.

Objeto asociado a una tabla. Desencadenará una acción cuando ocurra una de estas 3 cosas: Insertar (insert), actualizar (update) o eliminar(delete).

```
CREATE TABLE REG_PRODUCTOS (CÓDIGOARTÍCULO VARCHAR(25),  
NOMBREARTÍCULO VARCHAR(30), PRECIO INT(4), INSERTADO DATETIME)
```

```
CREATE TRIGGER PRODUCTOS_AI AFTER INSERT ON PRODUCTOS FOR EACH ROW  
INSERT INTO REG_PRODUCTOS  
(CÓDIGOARTÍCULO,NOMBREARTÍCULO,PRECIO,INSERTADO)  
VALUES (NEW.CÓDIGOARTÍCULO,NEW.NOMBREARTÍCULO,NEW.PRECIO,NOW())
```

NOTA: For each (row/statement) puede llevar uno de esos 2 predicados. Row para que se desencadene la acción cada vez que se inserta un registro, pero como en una misma sentencia se podrían insertar varios registros, se puede cambiar row por statement, para que se ejecute el trigger por cada sentencia.

```
Introducimos un producto: INSERT INTO PRODUCTOS  
(CÓDIGOARTÍCULO,NOMBREARTÍCULO,PRECIO,PAÍSDEORIGEN)  
VALUES('AR75','PANTALÓN',50,'ESPAÑA')
```

```
CREATE TABLE PRODUCTOS_ACTUALIZADOS(ANTERIOR_CÓDIGOARTÍCULO  
VARCHAR(4), ANTERIOR_NOMBREARTÍCULO VARCHAR(25), ANTERIOR_SECCIÓN  
VARCHAR(15), ANTERIOR_PRECIO INT(4), ANTERIOR_FECHA DATE,  
ANTERIOR_IMPORTADO VARCHAR(15), ANTERIOR_PAÍSDEORIGEN VARCHAR(15),
```

NUEVO_CÓDIGOARTÍCULO VARCHAR(4), NUEVO_NOMBREARTÍCULO VARCHAR(15),
NUEVO_SECCIÓN VARCHAR(15), NUEVO_PRECIO INT(4), NUEVO_FECHA DATE,
NUEVO_IMPORTADO VARCHAR(15), NUEVO_PAÍSDEORIGEN VARCHAR(15), USUARIO
VARCHAR(20), F_MODIF DATETIME)

```
CREATE TRIGGER PRODUCTOS_BU BEFORE UPDATE ON PRODUCTOS FOR EACH  
ROW INSERT INTO PRODUCTOS_ACTUALIZADOS (ANTERIOR_CÓDIGOARTÍCULO,  
ANTERIOR_SECCIÓN, ANTERIOR_NOMBREARTÍCULO, ANTERIOR_PRECIO,  
ANTERIOR_FECHA, ANTERIOR_IMPORTADO, ANTERIOR_PAÍSDEORIGEN,  
NUEVO_CÓDIGOARTÍCULO, NUEVO_SECCIÓN, NUEVO_NOMBREARTÍCULO,  
NUEVO_PRECIO, NUEVO_FECHA, NUEVO_IMPORTADO, NUEVO_PAÍSDEORIGEN,  
USUARIO, F_MODIF)  
VALUES  
(OLD.CÓDIGOARTÍCULO, OLD.SECCIÓN, OLD.NOMBREARTÍCULO, OLD.PRECIO, OLD.FE  
CHA, OLD.IMPORTADO, OLD.PAÍSDEORIGEN, NEW.CÓDIGOARTÍCULO, NEW.SECCIÓN, N  
EW.NOMBREARTÍCULO, NEW.PRECIO, NEW.FECHA, NEW.IMPORTADO, NEW.PAÍSDEORI  
GEN, CURRENT_USER(), NOW());
```

Un ejemplo: UPDATE PRODUCTOS SET PRECIO=65, SECCIÓN='CERÁMICA' WHERE
CÓDIGOARTÍCULO='AR75'

```
CREATE TABLE PROD_ELIMINADOS (C_ART VARCHAR(5), NOMBRE  
VARCHAR(25), PAIS_ORIGEN VARCHAR(15), PRECIO DECIMAL, SECCIÓN  
VARCHAR(15))
```

```
CREATE TRIGGER PRODUCTOS_AD AFTER DELETE ON PRODUCTOS FOR EACH ROW  
INSERT INTO PROD_ELIMINADOS (C_ART, NOMBRE, PAIS_ORIGEN, PRECIO, SECCIÓN)  
VALUES (OLD.CÓDIGOARTÍCULO, OLD.NOMBREARTÍCULO,  
OLD.PAÍSDEORIGEN, OLD.PRECIO, OLD.SECCIÓN)
```

DELETE FROM PRODUCTOS WHERE CÓDIGOARTÍCULO='AR41'

MODIFICAR TRIGGER: Primero hay que modificar la tabla donde se aloja la acción del trigger.

```
ALTER TABLE PROD_ELIMINADOS ADD COLUMN (USUARIO VARCHAR(15),  
FECHA_MODIF DATETIME)
```

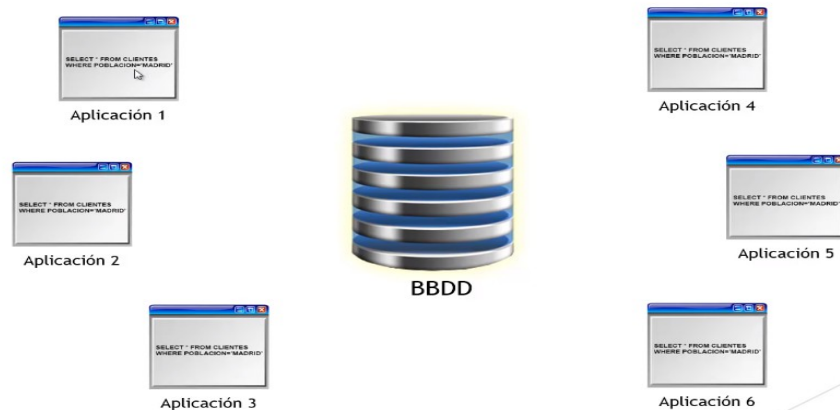
Luego eliminamos el trigger: DROP TRIGGER IF EXISTS PRODUCTOS_AD
Luego lo volvemos a crear incluyendo los nuevos campos:

```
CREATE TRIGGER PRODUCTOS_AD AFTER DELETE ON PRODUCTOS FOR EACH ROW  
INSERT INTO PROD_ELIMINADOS (C_ART, NOMBRE, PAIS_ORIGEN, PRECIO, SECCIÓN,  
USUARIO, FECHA_MODIF) VALUES (OLD.CÓDIGOARTÍCULO,  
OLD.NOMBREARTÍCULO, OLD.PAÍSDEORIGEN, OLD.PRECIO, OLD.SECCIÓN, USER(),  
NOW())
```

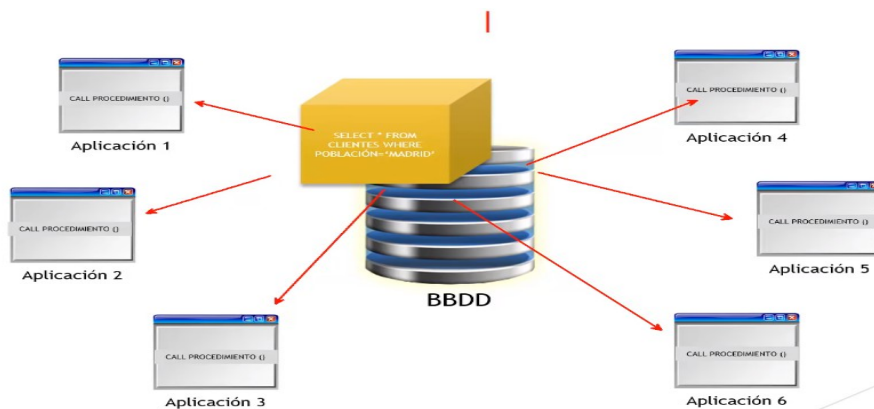
PROCEDIMIENTOS ALMACENADOS:

- ¿Qué es un procedimiento almacenado? Utilidad de un PA
- Ejemplos de PA sencillos

Procedimiento almacenado



Por cuestiones de eficiencia y seguridad se almacenan instrucciones que se usan mucho en procedimientos almacenados. Es más seguro porque el cliente no puede hacer inyecciones SQL, ni modificar el código SQL porque sólo permitimos que llamen a un procedimiento que está encapsulado.



```
CREATE PROCEDURE MUESTRA_CLIENTES_MADRID() SELECT * FROM CLIENTES WHERE POBLACIÓN='MADRID'
```

Se llama así: `CALL MUESTRA_CLIENTES_MADRID();`

```
CREATE PROCEDURE ACTUALIZA_PRODUCTOS(N_PRECIO DECIMAL(10,2), CODIGO VARCHAR(4)) UPDATE PRODUCTOS SET PRECIO=N_PRECIO WHERE CÓDIGOARTÍCULO=CODIGO;
```

```
CALL ACTUALIZA_PRODUCTOS(146.25, 'AR40')
```

Para mostrar los procedimientos que tenemos: [SHOW PROCEDURE STATUS](#) o en la pestaña de rutinas de la BBDD o en el árbol de navegación de la BBDD aparece un nuevo desplegable llamado Procedimientos.

PROCEDIMIENTOS Y TRIGGERS (TRIGGERS CONDICIONALES):

- Declaración de variables en procedimientos almacenados.
- Triggers condicionales.

Declareremos una variable en el procedimiento que será la edad:

Usaremos como delimitador de bloque \$\$ lo establecemos con la siguiente instrucción se suele utilizar también //):

Ejemplo1:

```

1 DELIMITER $$
2
3 CREATE PROCEDURE CALCULA_EDAD(AGNO_NACIMIENTO INT)
4
5 BEGIN
6
7     DECLARE AGNO_ACTUAL INT DEFAULT 2016;
8
9     DECLARE EDAD INT;
10
11     SET EDAD=AGNO_ACTUAL-AGNO_NACIMIENTO;
12
13     SELECT EDAD;
14
15 END;$$
16
17 DELIMITER ;
18

```

Ejemplo2:

DELIMITER \$\$

```

CREATE PROCEDURE CALCULA_EDAD(FECHA_NACIMIENTO DATE)
BEGIN
    DECLARE EDAD INT;
    SET EDAD=DATEDIFF(CURDATE(),FECHA_NACIMIENTO)/365.25;
    SELECT EDAD;
END;$$
DELIMITER ;

```

Con la última instrucción indicamos que el delimitador por defecto vuelve a ser el ; . No es necesaria está última sentencia.

```
CALL CALCULA_EDAD('1977-05-11')
```

☐ Mostrar todo | Número de filas: 25 ▼ | Filtrar filas:

+ Opciones

EDAD

44

Un ejemplo con Triggers:

```


DELIMITER $$
CREATE TRIGGER REVISA_PRECIO_BU BEFORE UPDATE ON PRODUCTOS FOR EACH
ROW
BEGIN
    IF(NEW.PRECIO<0 THEN SET NEW.PRECIO=0;
    ELSEIF (NEW.PRECIO>1000) THEN SET NEW.PRECIO=1000
    END IF;
END;$$
DELIMITER ;

```

Hemos probado a introducirle desde PHPMYADMIN un precio de 1200 euros al artículo AR40 (BOTA ALPINISMO) y el resultado que obtenemos es, una vez actualizado, un valor de 1000 euros para el campo precio. Esto indica que el trigger que hemos creado como procedimiento almacenado está funcionando.

También probamos que no admite precios negativos:

[UPDATE](#) productos [set](#) precio=-500 WHERE CÓDIGOARTÍCULO='AR40 '

<input type="checkbox"/>	 Editar  Copiar  Borrar	AR40	DEPORTES	BOTA ALPINISMO	0.00	2002-05-05	FALSO	ESPAÑA	NULL
--------------------------	--	------	----------	----------------	------	------------	-------	--------	------

VISTAS:

- ▶ ¿Qué son las vistas?
- ▶ Creación de vistas
- ▶ VENTAJAS:
 - ▶ Privacidad de la información
 - ▶ Optimización de la BBDD
 - ▶ Entorno de pruebas

Privacidad: Se pueden crear roles que tengan acceso a vistas y no a las tablas.

Optimización: Cuando se repiten las mismas consultas siempre, se consultarían las vistas en vez de generar las consultas.

```

CREATE VIEW ART_DEPORTES AS
SELECT NOMBREARTÍCULO,SECCIÓN,PRECIO FROM PRODUCTOS
WHERE SECCIÓN='DEPORTES'

```

```

CREATE VIEW ART_CERAMICA AS
SELECT NOMBREARTÍCULO,SECCIÓN,PRECIO FROM PRODUCTOS
WHERE SECCIÓN='CERÁMICA'

```



```
DROP VIEW ART_CERAMICA;
```

```
ALTER VIEW ART_DEPORTES AS  
SELECT NOMBREARTÍCULO,SECCIÓN,PAÍSDEORIGEN FROM PRODUCTOS  
WHERE PAÍSDEORIGEN='ESPAÑA'
```

Para dejarla como estaba porque ahora no tiene mucho sentido ese nombre:

```
ALTER VIEW ART_DEPORTES AS  
SELECT NOMBREARTÍCULO,SECCIÓN,PRECIO FROM PRODUCTOS  
WHERE SECCIÓN='DEPORTES'
```

Muy importante: Las vistas se actualizan automáticamente cuando hacemos modificaciones en la tabla origen de la vista.