

Consultas de acción

Comandos para ACTUALIZAR:

* **Actualizar los CAMPOS de tipo precio, incrementando el precio.**



* **Actualizar nombre de REGISTRO.**



Comandos para CREAR:

**- Crear BASE DE DATOS:** CREATE DATABASE *nombre*;

**- Crear TABLA:**

1. Datosusuarios: Nombre tabla.

2. Nombre: Nombre del primer campo.

3. Varchar: Especificamos el tipo de datos que se guarda (tipo texto de 30 o 10 caracteres en este caso).

4. Clave: Nombre del segundo campo.

Comandos para ELIMINAR:

* **Borrar TABLA:**



* **Borrar BASE DE DATOS:**



* **Borrar REGISTRO de una tabla:**



* **Borrar REGISTRO con OPERADORES LÓGICOS:**



Comandos para AGREGAR / INSERTAR:

* **Agregar CAMPO a una tabla:**



* **Agregar/Insertar DATOS dentro de cada campo:**



Comandos para ALTERAR:

* **Agregar CAMPO en una tabla:**



* **Borrar CAMPO de una tabla:**



* **Alterar tipo de dato en REGISTRO:**



* **Establecer valor por defecto (en vez de NULL)**





* + Y para borrarlo:



* **Indicar CLAVE PRIMARIA el crear una tabla:**



* + Indicar clave primaria con la tabla ya creada:



Comandos de Acces (SELECT INTO):

* **Crear tabla COPIANDO datos de otra tabla (solo en Acces)**





* **Lo mismo, pero en MySQL (CREATE TABLE):**



Comandos para VER / SELECT:

* **Ver información dentro ALGUNOS campos:**



* **Ver información de TODOS los campos:**



* **Ver información de dos REGISTROS (OR, AND) dentro de una tabla:**





* **Ver registros que estén comprendidos ENTRE DOS fechas:**



* **Ver registros de dos secciones, ORDENADOS:**
  + *Orden ascendente:*



* + *Orden descendente:*
  + *Ordenar por dos registros:*



Comandos Datos Anexados:

* + *Nos va a permitir, anexar/unir los datos de una tabla, en otra.*
* **Anexar/Unir toda la tabla de CLIENTE\_MADRID, dentro de la tabla CLIENTES:**

**(En Acces y MYSQL)**

* **Lo mismo que lo anterior, pero especificando los campos que queremos anexar.**
  + Sin la DIRECCIÓN y RESPONSABLE



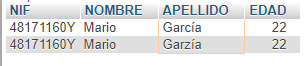
Comandos usando LIKE:

* **Encontrar apellidos LIKE Gómez.**

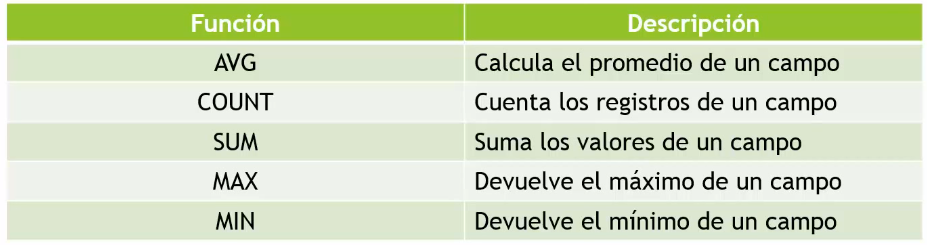


* **Si alguien se equivoca y en vez de poner GarCía, ponen GarZia, la forma de descartar esa letra para encontrar los resultados es la siguiente:**





Consultas por agrupación



Función SUM (Suma)

1. **(SUM) Agrupar campos y sumar el precio de las agrupaciones:**





* 1. **(SUM) Ordenar lo anterior por precio.**
  + Si ejecutamos lo siguiente dará error:



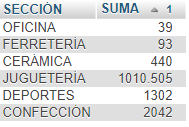
Debido a que en la tabla resultante no tenemos ningún campo que se llame precio. Y lo que hace la cláusula ORDER BY es buscar el registro en la consulta, no en la tabla.

* + - Consulta:



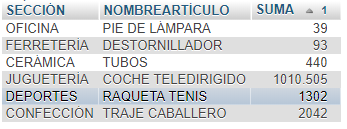
Se puede solucionar utilizando un **ALIAS,** que nos permite cambiar el nombre a un campo. Le cambiamos el nombre a **SUM(PRECIO)** por **SUMA**, y aplicamos el orden a **SUMA**.





* 1. **(SUM) Además de lo anterior, queremos que nos aparezca el primer elemento de cada campo (de la tabla).**





Función AVG (Promedio)

1. **(AVG) Sacar la media del precio de todos los registros que tengan como campo SECCIÓN el nombre registro de: DEPORTES o CONFECCIÓN:**
   * HAVING sustituye a WHERE en consultas de agrupación.



Función COUNT (Contar)

1. **(COUNT) Cuantos clientes tengo por cada provincia:**
   * La función COUNT solamente cuenta los registros de los campos que están rellenados.

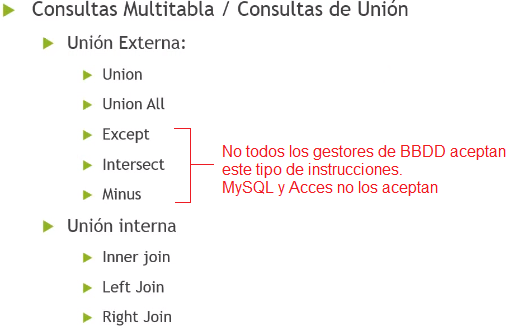


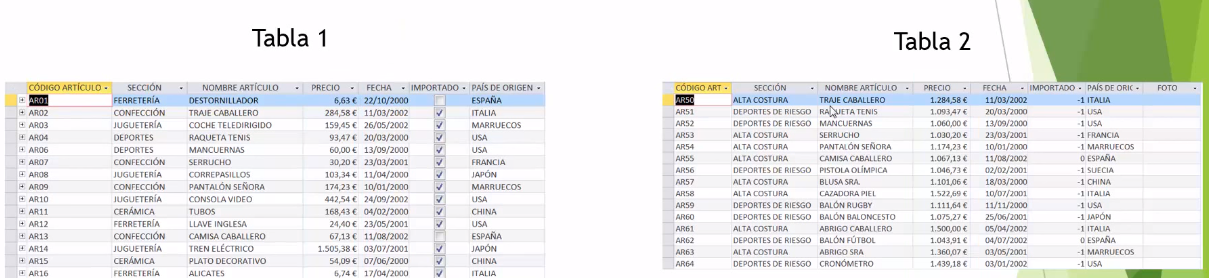
Función MAX (Máximo)

1. **(MAX) Ver el precio más alto entre todos los productos cuya SECCIÓN sea CONFECCIÓN:**



Consultas multitabla



* Deben de tener campos compatibles y el mismo número de campos.
* El campo PRECIO, por ejemplo, no puede ser de tipo VARCHAR en una tabla, y en la otra de tipo numérico.

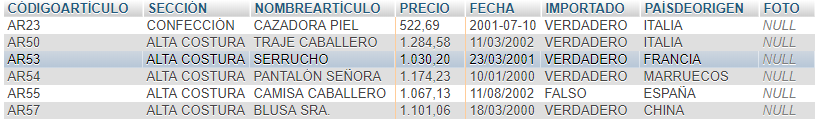
Consultas multitabla (UNIÓN EXTERNA)

* + **UNION:** En caso que hay registros repetidos en taba 1 y tabla 2, esos registros solo los muestra una vez.
  + **UNION ALL:** En caso de haber registros repetidos en tablas, los muestra tantas veces como estén contemplados en las tablas.

1. **(UNION) Buscar registros dentro de dos campos iguales (SECCIÓN) en dos tablas distintas:**



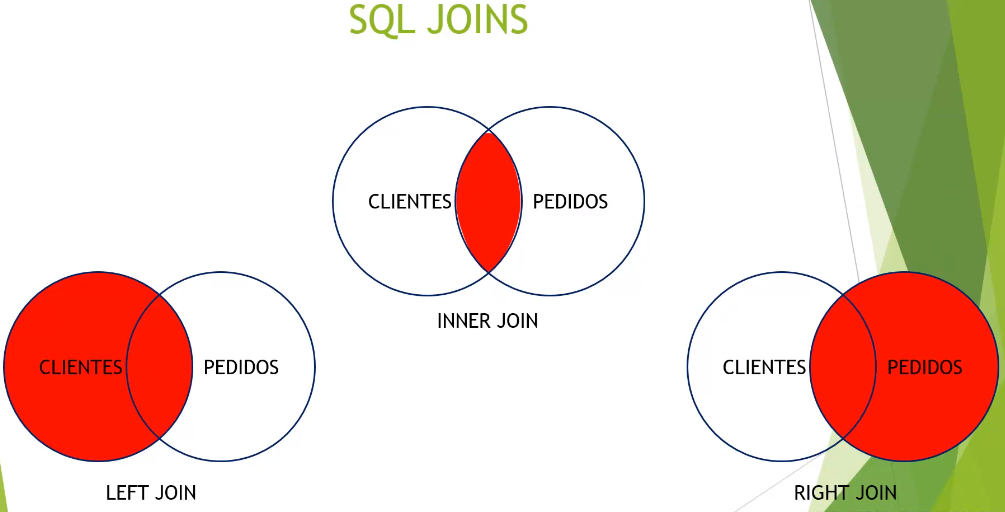
* 1. **(UNION) Buscar registros de una tabla cuyo precio sea mayor a 500, y todos los productos de otra tabla cuya sección sea ALTA COSTURA.**



1. **(UNION ALL) Buscar todos los registros de dos tablas teniendo en cuenta los registros repetidos (para que se muestren).**



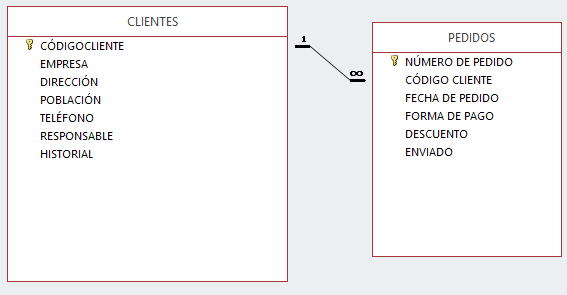
Consultas multitabla (UNIÓN INTERNA)

**SQL JOINS: Nos devuelve registros que se contemplan en tos tablas.**

* + **INNER JOIN:** Nos devuelve únicamente los registros que están contemplados en dos tablas.
  + **LEFT JOIN:** Nos devuelve los registros contemplados en las dos tablas, más los que están ‘en la tabla de la izquierda’.
  + **RIGHT JOIN:** Nos devuelve los registros contemplados en las dos tablas, más los que están ‘en la tabla de la derecha’.

1. **(INNER JOIN) Buscar clientes de MADRID que hayan hecho pedidos.**
   * Hay que tener en cuenta cual es el campo que relaciona cada tabla:

(En este caso CÓDIGO CLIENTE).



También hay que tener en cuenta que en una tabla CÓDIGO CLIENTE se escribe junto, y en la otra no, por eso hay que escribir el que va separado entre [ ].

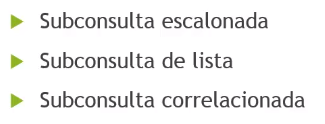
* + *Traducción del código:*

*Seleccionar todos desde* ***Clientes -> INNER JOIN <-*** *Pedidos donde el campo* ***CÓDIGOCLIENTE*** *(de clientes) sea igual que* ***CÓDIGO CLIENTE****, donde la población sea Madrid.*

1. **(LEFT JOIN) Buscar clientes de MADRID que hayan hechos pedidos o NO (teniendo en cuenta que aparecerán todos los clientes, hayan hecho pedidos o no).**
   * *Si nos fijamos, CT19, no ha hecho ningún pedido, pero aun y así, sale en los resultados*
2. **(RIGHT JOIN) Buscar clientes de MADRID que hayan hechos pedidos o NO (teniendo en cuenta la tabla de pedidos, cosa que no se puede realizar ya que no se pueden mostrar los pedidos de las personas que no los han hecho).**

Subconsultas

*Consultas dentro de otra consulta*

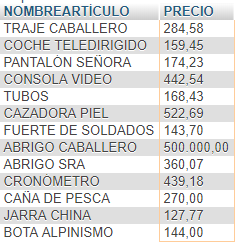


Subconsulta escalonada

La consulta interna devuelve una única columna con un único registro y eso se utiliza como único criterio o comparación en la consulta padre:

1. **Ver NOMBREARTÍCULO, SECCIÓN de los productos que su precio sea mayor a: *El precio de media que tienen todos los productos de la tabla.***

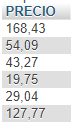




Subconsulta de lista

La consulta interna devuelve una lista de registros (suele utilizar IN, ANY, ALL):

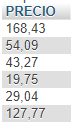
1. **(ALL) Mostrar los artículos de la tabla de PRODUCTOS cuyo precio sea superior a TODOS los artículos de CERÁMICA.**



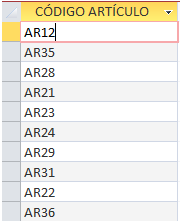


1. **(ANY) Mostrar los artículos de la tabla de PRODUCTOS cuyo precio sea superior CUALQUIERA los artículos de CERÁMICA.** 
   * *(Es decir, los productos que tengan más precio en comparación al que tiene el precio menor entre los productos de CERÁMICA)*



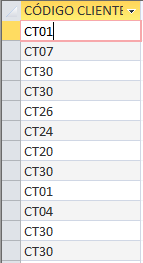


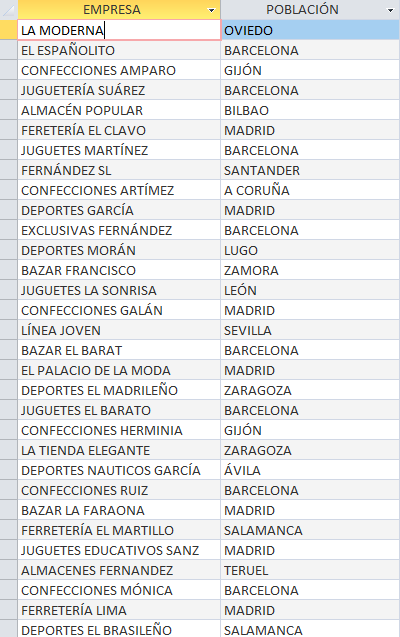
1. **(IN) Mostrar NOMBREARTICULO + PRECIO de la tabla PEDIDOS, de los productos que tengan más de 20 unidades en la tabla PRODUCTOS- PEDIDOS.**





* + *Consulta anterior con consulta multitabla INNER JOIN:*
  1. **(NOT IN) Mostrar CLIENTES que NO han hecho pedidos con tarjeta o que NO han hecho pedidos.**



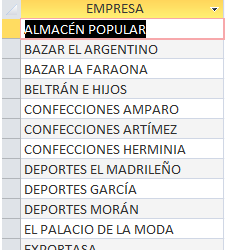


Predicados

* + **DISTINCT:** Poniéndolo en la consulta; delante de un campo que no queremos que se repita.
  + **DISTINCTROW:** Poniéndolo en la consulta; descarta las filas que sean que estén repetidas.

1. **(DISTINCT) Saber los CLIENTES que han hecho pedidos (SIN REPETIR CAMPOS).**
   * *Consulta de INNER JOIN para saber los que han hecho pedidos:*

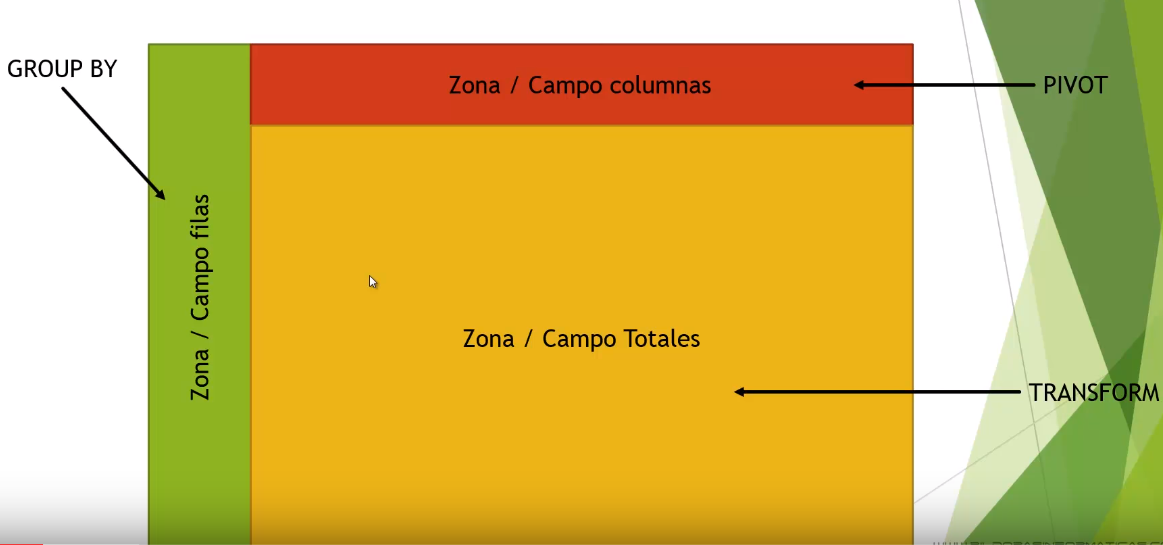


* + ***Consulta como la anterior, per* *o añadiendo DISTINCT para evitar repetidos.*

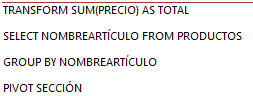
1. **(DISTINCTROW) Hacer consulta para mostrar registros de tipo FERRETERÍA, sin repetir filas:**
   * *Sin tener en cuenta los repetidos (sin DISTINCTROW):*



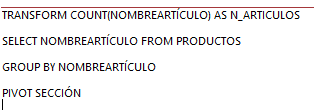
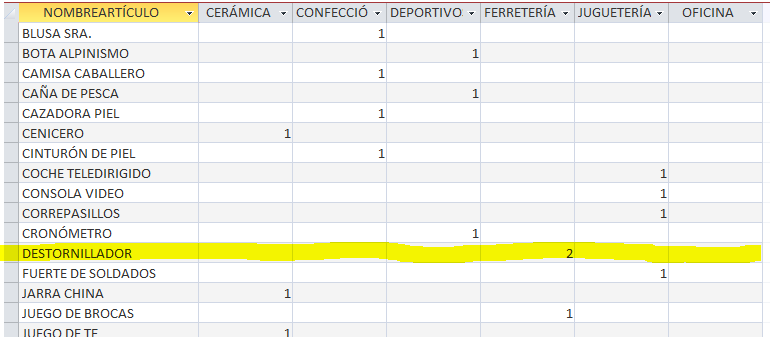
* + *Usando DISTINCTROW:*
  1. **(DISTINCTROW y LEFTJOIN) Borrar todos los CLIENTES que no hayan hecho PEDIDOS.**
  + *Se hace uso de la función LEFT JOIN (para crear una tabla con los CLIENTES que hayan hecho PEDIDOS, y los que no)*

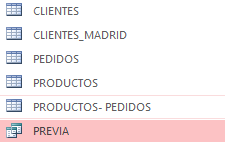
Referencias cruzadas

1. **Crear una tabla con referencias cruzadas para saber PRECIOS de artículo.**
   * En la zona de columnas (**PIVOT**) agrupamos todas las SECCIONES.
   * En la zona de filas (**GROUP** **BY**) agrupamos los NOMBREARTÍCULO.
   * En la zona de totales (**TRANSFORM**) agrupamos **la suma de los precios** de los NOMBREARTÍCULOS (por si acaso hay más de uno, como, por ejemplo, DESTORNILLADOR, que hay 2).



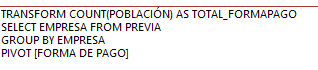


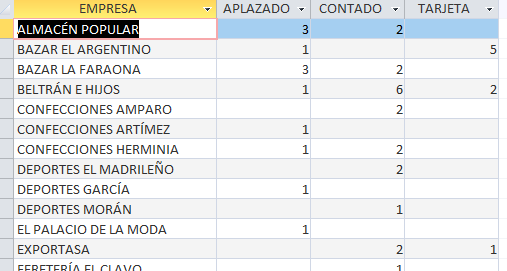
* 1.  **Crear una tabla con referencias cruzadas para saber UNIDADES de cada artículo.**
  + *Vemos que tenemos 2 unidades de DESTORNILLADOR.*
  1. **Contar los pagos que nos han hecho los clientes, y como los han hecho.**
  + Anteriormente hemos hecho una consulta de tipo INNER JOIN y se ha guardado la tabla con el nombre PREVIA.



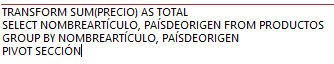


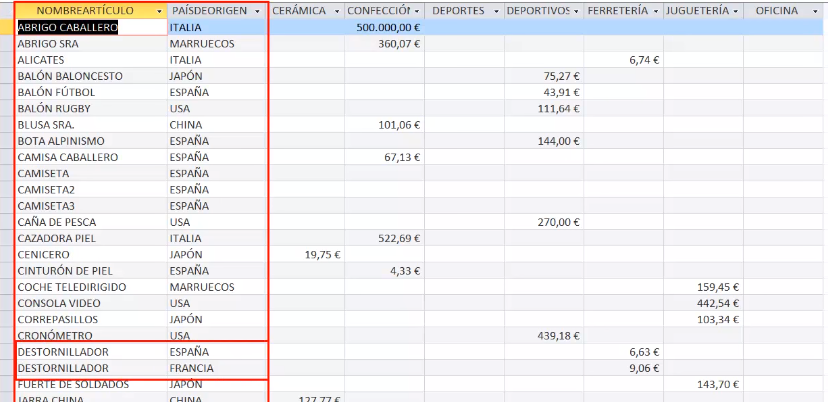


* + Y ejecutamos la consulta basándonos en la tabla PREVIA:

****

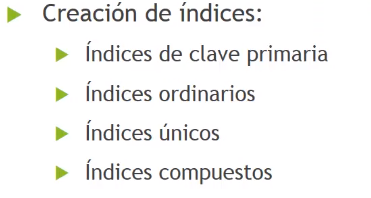
* 1. **Hacer consulta para que nos aparezcan dos campos en la zona de filas.**
  + En este caso sí que nos aparecerán dos registros de tipo DESTORNILLADOR, ya que cada uno es de un país distinto.

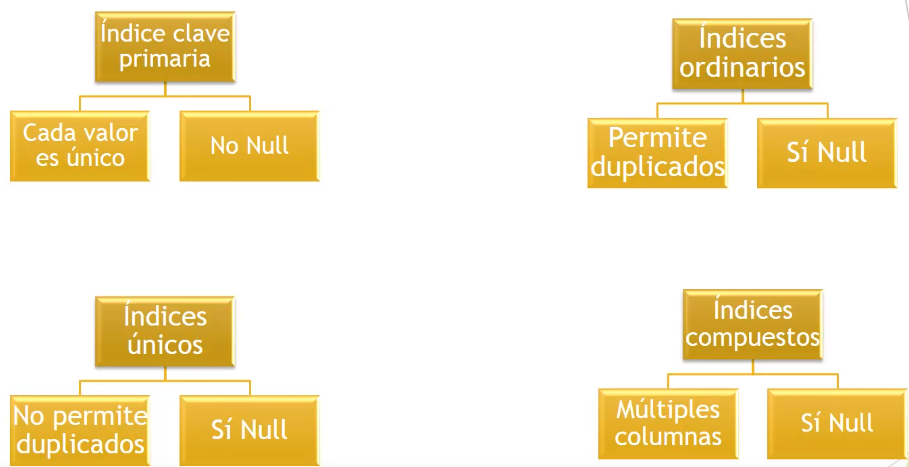




Referencias

* + Nos permiten hacer búsquedas más rápidas en una tabla de una BBDD.
  + No forman parte del estándar de SQL (cada gestor de BBDD tiene su propio comando).





* **Indicar CLAVE PRIMARIA el crear una tabla:**

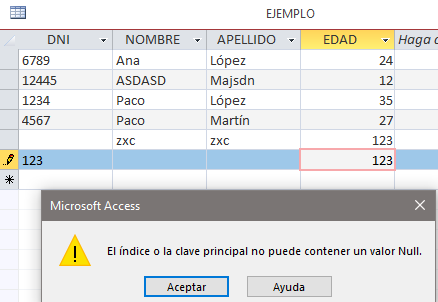


* + Indicar clave primaria con la tabla ya creada:

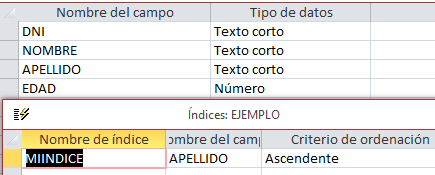


Índice clave primaria

* + No permite dejar un índice vacío y tampoco repetir los dos índices:
* **Dos claves, NOMBRE y APELLIDO (No podemos dejar los índices vacíos y tampoco, podemos repetirlos (No pueden haber 2 Paco López)).**



Índice ordinario

* + Permite duplicado y permite dejar en blanco el CAMPO.
* **Crear índice, especificando cuál es el REGISTRO que queremos que tenga el índice:**



Índice único

* + Permite duplicado y permite dejar en blanco el CAMPO.
* **Crear índice, especificando cuál es el REGISTRO que queremos que tenga el índice:**



Índice compuesto

* + Es el que está compuesto por 2 o más campos y permite dejar en blanco el CAMPO. No permitirá haber 2 personas con el mismo nombre y apellido.
* **Crear índice, especificando cuál es el REGISTRO que queremos que tenga el índice:**

Eliminar índice y clave primaria

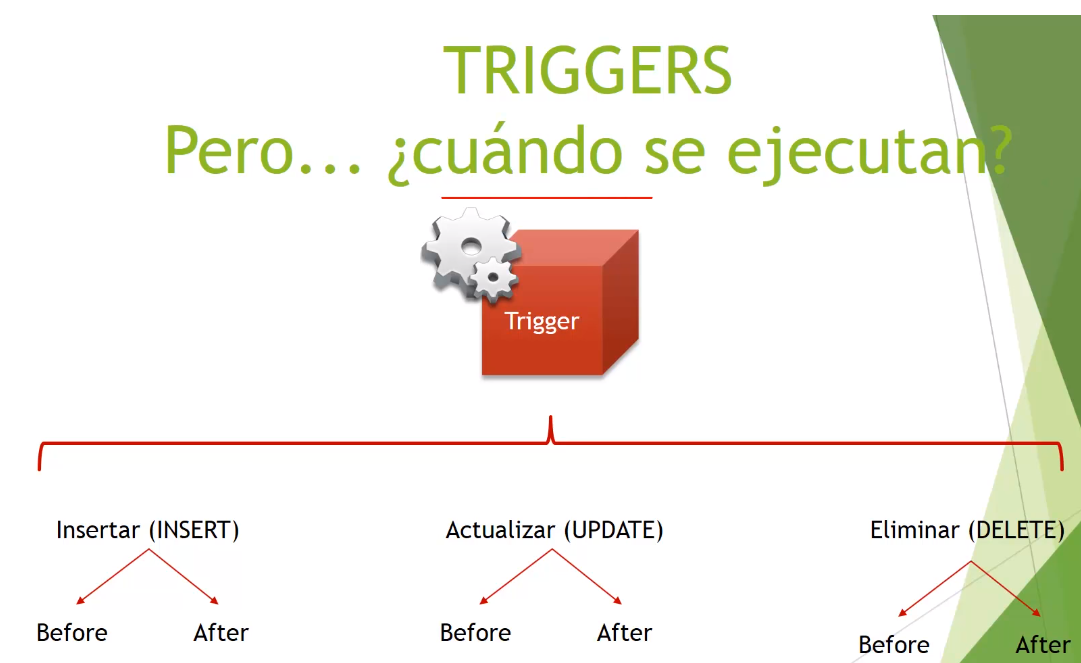
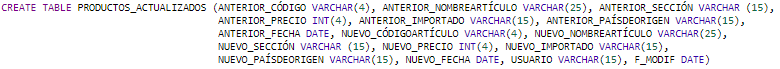
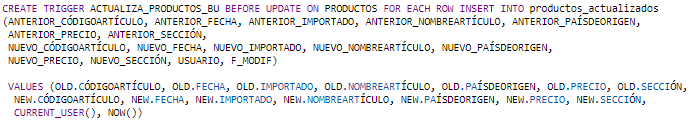
* **En Acces, SQL Server, Oracle:**



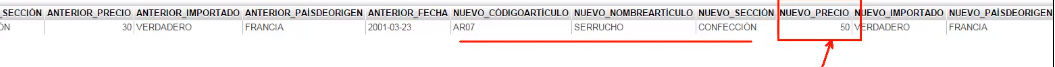
* **En MYSQL:**



Triggers (Disparadores)

* + Desencadena una acción cuando se hacen modificaciones de tipo INSERT, UPDATE, ELIMINAR.
  + En Acces no se pueden utilizar.
* **(TRIGGER INSERT) Cuando insertamos un nuevo producto en la tabla PRODUCTOS, insertar el mismo producto (especificando los campos a rellenar de la nueva tabla) en otra tabla llamada REG\_PRODUCTOS.**
  + **FOR EACH ROW:** Por cada fila que insertemos, se inserta otra, en la tabla reg\_productos.
  + **NOW():** Función que crea la fecha y la hora en la que se ha ejecutado la acción (aparecerá: )
* **(TRIGGER UPDATE) Al hacer una modificación en un campo de la tabla, ejecutar un trigger BEFORE para que cree una tabla alternativa de respaldo para saber el nuevo valor del campo actualizado, y el antiguo.**
  + Se crea la tabla:
  + Se crea el trigger:
  + Se ejecuta el comando para actualizar:



* + Resultado:

* **(TRIGGER DELETE) Cuando se borra una fila en una tabla, hacer una copia de seguridad de esa fila y guardarla en otra tabla (PROD\_ELIMINADOS)**
  + Se crea la tabla:
  + Se crea el trigger:



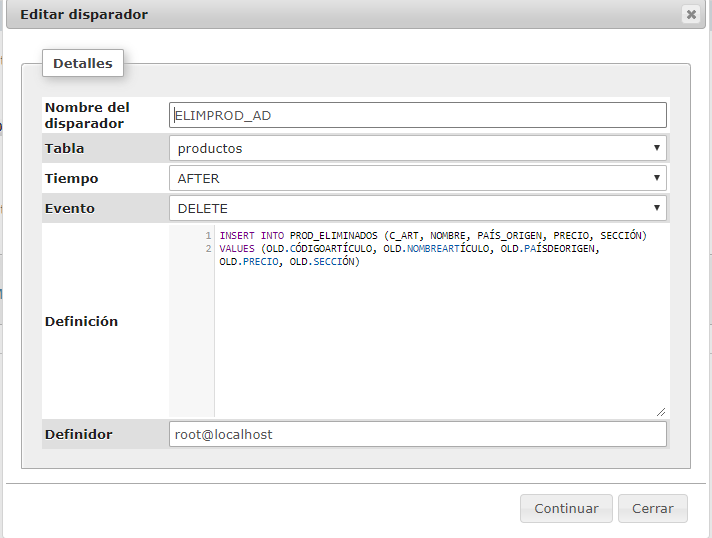
* + Se ejecuta la instrucción:



* + Resultado:



Modificar trigger



Procedimientos almacenados

* + Creamos prodecimiento:



* + Si lo llamamos, ejecuta lo que le hemos pedido que haga, al procedimiento:



Consultas con procedimientos con pase de parámetros

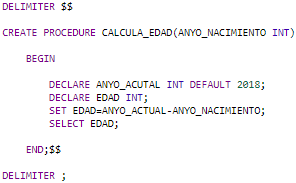
* + Creamos procedimiento:



* + Llamamos al procedimiento pasándole parámetros:



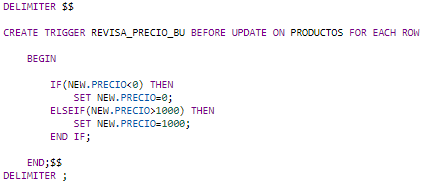
* **Procedimientos, con delimitadores:**
  + Creamos el procedimiento, especificando delimitador de bloque ($$), y por cada declaración de variable añadimos ‘;’. Luego especificamos al final que el delimitador de línea será ‘;’.



* + Ejecutamos la consulta y el resultado es correcto:



* **Procedimientos, con delimitadores y condicionales:**
  + Crear procedimiento, que al modificar el precio de los productos, aplica un condicional (en el caso que el precio sea menor que 0, no pintar en la tabla un número negativo, y en el caso de que sea superior a 1000, que aparezca solo el número 1000.



* + Ejecutamos la modificación y el resultado es correcto:





* + Modificación de precio que supera los 1000:



Vistas

* + Privacidad de la información.
  + Optimización de la BBDD.
  + Entorno de pruebas (para desarrolladores, para comprobar que datos existen en la BBDD, sin peligrar los datos)
* **Crear vista para ver campos de la SECCIÓN DEPORTES en la tabla PRODUCTOS:**
  + Creación VISTA:



* + Resultado:



* + Si se hace modificación en la tabla principal, en la SECCIÓN DEPORTES, también se verá reflejada la modificación en la vista.
* **Eliminar VISTA:**
* **Modificar VISTA:**