## COMPONENTES DE LAS BASES DE DATOS:

* Base de datos
* Tabla
* Campo
* Registro
* Relación
* Vista
* Índice
* Trigger
* Función
* Procedimiento
* Evento
* Schema
* Logfile group
* Server
* Tablespace
* Tipos de datos SQL
  + Numéricos
    - Enteros
      * TINYINT (1 bit, Rango desde el -128 hasta el 127 SIGNED, desde el 0 hasta el 255 UNSIGNED)
      * SMALLINT (2 bit, de -32768 a 32767 SIGNED, 0 a 65.535 UNSI)
      * MEDIUMINT (3 bit, -8 mill a 8 mill, 0 a 17 mill)
      * INTEGER (4 bit, -2000 a 2000 mill, de 0 a 4.000 millones)
      * BIGINT (8 bit, -10 trillones a +10trillones, 20 trillones)
    - Decimales
      * DECIMAL (M,D): Valor fijo. Admite M caracteres en total, de lo cuales D serán la parte decimal.
      * NUMERIC (M,D): Sinónimo de DECIMAL
      * FLOAT (M,D): Tiene rango alto, admite números grandes de muchos dígitos pero da un valor aproximado (redondea)
      * DOUBLE(M,D): Como FLOAT (redondea)
    - Boolean
  + Cadenas de caracteres
    - CHAR (M): Tiene longitud fija M, si fuese más corta se rellena con espacios, si fuese más larga daría error o se cortaría, ocupará M bites, el máximo de este tipo de cadena son 255 caracteres.
    - VARCHAR (M): Tiene longitud variable limitado a M caracteres. Ocupa M bites + 1.
    - TINYTEXT: Ocupa M+1 bit. Máximo 255 caracteres. No permite default.
    - TEXT: Ocupa M+2 bit. Máximo 65.000 caracteres. No permite default.
    - MEDIUMTEXT: Ocupa M+3 bit. Máximo 16 millones de caracteres. No permite default.
    - LONGTEXT: Ocupa M+4 bits. Máximo 4.000 millones de caracteres. No permite default.
    - ENUM: Solo admitirá los valores que se incluyan como argumentos (se pone la lista entre paréntesis al declarar ENUM.
  + Fechas
    - DATE: Formato “YYYY-MM-DD”. Rango “1000-01-01” – “9999-12-31”
    - TIME: Formato “HH:MM:SS”. Rango “-838:59:59”-“838:59:59”
    - DATETIME: “YYYY-MM-DD HH:MM:SS”. Rango “1000-01-01 00:00:00” – “9999-12-31 23:59:59”
    - TIMESTAMP: Es como datetime pero guarda la hora local y cuando se recupera lo hace mostrando la hora local del momento en que se guardó el dato, no la hora del sitio en el que se introdujo el dato en su momento, así que puede mostrar distintos valores cada vez que se consulte, según dónde se esté. Formato “YYYY-MM-DD HH:MM:SS”. Rango “1970-01-01 00:00:01” UTC – “2038-01-19 03:14:07” UTC.
    - YEAR: Formato “YYYY” o “YY”. Rango “1901” – “2155” o (19)“70” – (20)”69”.
* Componentes del lenguaje SQL
  + COMANDOS
    - DDL
      * CREATE
      * DROP
      * ALTER
      * TRUNCATE
    - DML
      * SELECT
      * INSERT
      * DELETE
      * UPDATE
      * CALL
    - DCL
      * GRANT
      * REVOKE
      * COMMIT
      * ROLLBACK
    - OTROS
      * DECLARE
      * OPEN
      * FLETCH
      * CLOSE
  + CLÁUSULAS
    - FROM
    - GROUP BY
    - HAVING
    - ORDER BY
    - WHERE
    - LIMIT
  + OPERADORES LÓGICOS
    - AND
    - OR
    - NOT
    - XOR
    - Is
    - Eqv
    - Imp
  + OPERADORES DE COMPARACIÓN
    - <
    - >
    - <>
    - >=
    - <=
    - BETWEEN
    - LIKE
    - IN
  + FUNCIONES DE AGREGADO
    - AVG
    - COUNT
    - SUM
    - MAX
    - MIN
  + OTROS TEMAS
    - TIPOS DE VALORES
    - OTRAS PALABRAS QUE NO SÉ LO QUE SON
      * TOP
      * JOIN
      * OFFSET
      * SET
      * MERGE
      * PERCENT
      * WITH TIE
      * UNION
      * UNION ALL
      * AS
      * INNER JOIN
      * DESC
      * ASC
      * OUTPUT
      * INTO
      * SHOW
    - Comodines de búsqueda
      * %
      * \_ (guion bajo) (sustituye a un carácter cualquiera, cada guion a un carácter. Si pongo 3 guiones serían 3 caracteres).
    - Comentarios:

Hay 3 formas: /\* y \*/ para comentarios de varias líneas, -- para comentarios de una línea y # para comentarios de una línea.

* + - * /\* Esto es un comentario

de varias líneas\*/

* + - * -- Comentario de una línea.
      * # Comentario de una línea.

## EXPORTAR BASE DE DATOS DESDE MYSQL (IDB A SQL):

1. Se hace desde el símbolo del sistema, no desde la línea de comandos. Hay que abrir el símbolo del sistema COMO ADMINISTRADOR. Al ejecutarlo como administrador aparecerá normalmente C:/Windows/system32>

2. Con el explorador de Windows vamos a la carpeta donde está el archivo que hace el “dump” que en mi caso es mysqldump.exe, y que está en C:/Archivos de programa/MySQL/MySQL server 8.0/bin

3. Se copia esa ruta (porque es donde está ese archivo).

4. Nos vamos a esa ruta en el símbolo del sistema (poniendo cd y pegando la ruta y pulsando enter).

5. Una vez en esa carpeta en el símbolo del sistema, se entra en MySQL. Para ello, sin salir del símbolo del sistema se escribe lo siguiente: mysql –h localhost –u root –p

Y se pulsa enter. Entonces te pide tu contraseña de mysql. La metes.

6. Apareces en mysql>

Pues sin hacer nada ahí, vuelves a salir escribiendo quit y pulsando enter. Te vulve a llevar a la ruta anterior (a la carpeta bin)

7. Una vez que vuelves a estar en la ruta que copiaste y pegaste antes (carpeta bin) escribes el comando para exportar el archivo, que es: mysqldump –h localhost –u root –p nombrebasededatosquequieresexportar > nuevonombrebaseexportada.sql

Y ya exportó la base de datos a la carpeta bin.

## INSERTAR CAMPOS CONTRASEÑA:

## Password

Esta función está presente desde hace unas cuantas versiones, pero sin embargo se ha demostrado que es fácilmente vulnerable por lo que no merece la pena explicarla.

## MD5

Para guardar una contraseña encriptada con MD5 necesitaremos una tabla con un campo de 32 caracteres, aunque se ha demostrado que el algoritmo MD5 puede ser vulnerado, la práctica es tan compleja que no merece la pena el esfuerzo, el algoritmo MD5 no puede ser revertido, es decir, no se pueden recuperar contraseñas de este sistema.

**Insertar una contraseña con MD5:**  
mysql> INSERT INTO usuarios VALUES('usuario',MD5('contraseña'));

## SHA o SHA1

La función SHA y SHA1 son sinónimos, por lo que hacen el mismo efecto, al igual que MD5 no puede ser revertido y este necesita un campo de 40 caracteres para su almacenamiento, es más seguro que MD5 ya que calcula el cheksum SHA de 160 bits de una cadena, mientras que MD5 la calcula de 128.

**Insertar una contraseña con SHA:**  
mysql> INSERT INTO usuarios VALUES('usuario',SHA('contraseña'));

## AES

El algoritmo AES es el más completo y complejo, tiene posibilidad de revertirse y se utiliza con una llave privada, es decir, debemos proporcionar dicha llave para encriptar y desencriptar las contraseñas. Sería vulnerable en el caso de que alguien consiguiese violar nuestro código fuente y viera que llave estamos pasando a la cadena. Para almacenar estas contraseñas necesitamos que nuestro campo sea de tipo **BLOB**, ya que el resultado de la operación será un dato binario muy aleatorio.

**Insertar una contraseña con AES:**  
mysql> INSERT INTO usuarios VALUES('usuario',AES\_ENCRYPT('contraseña','llave'));  
**Para desencriptar:**  
mysql> INSERT INTO usuarios VALUES('usuario',AES\_DECRYPT('contraseña','llave'));

Si quieres utilizar la función AES, debes de tener una versión de MySQL superior a la 4.0.2, ya que es cuando esta función fue agregada.

Por lo tanto ahora sabemos que la única manera (por ahora) de que nuestras contraseñas sean totalmente seguras, es utilizando la función AES, y asegurarse que nadie puede acceder a nuestro código fuente.

## BIBLIOGRAFÍA:

Bibliografía: <https://snmb-admin.readthedocs.io/en/latest/geotalleres/conceptos-sql/conceptos_sql.html>

<https://docs.microsoft.com/es-es/sql/t-sql/queries/top-transact-sql?view=sql-server-ver15>

<https://gestionbasesdatos.readthedocs.io/es/latest/Tema1/index.html>

<https://www.bitdegree.org/learn/sql-commands-list#commit>

<https://virtual.itca.edu.sv/Mediadores/dbd/u2/23_clausula_alter.html>

## EJEMPLOS:

* En mysql> Para mostrar todas las bases de datos:
  + SHOW DATABASES;
* Para entrar en una base de datos para hacer cosas dentro de ella:
  + USE agenda;
* Para ver las tablas de una base de datos, una vez dentro de la BD:
  + SHOW tables;
* Crear base de datos:
  + CREATE DATABASE alumnos;
* Borrar base de datos (borra la base de datos por completo):
  + DROP DATABASE alumnos;
* Borrar una base de datos en caso de que exista:
  + DROP DATABASE IF EXISTS alumnos;
* Borrar base de datos (borra todo el contenido pero deja el archivo):
  + TRUNCATE DATABASE alumnos;
* Crear una tabla llamada amigos en una base de datos llamada agenda. Para el campo id\_amigo ponemos tipo de dato SMALLINT (permite hasta el 65.535), UNSIGNED porque no tendrá negativos, se pone NOT\_NULL aunque no sería necesario ya que va a ser la clave primaria y el campo clave primaria es declarado automáticamente por el programa como NOT\_NULL, lo ponemos AUTO\_INCREMENT para que lo rellene el programa automáticamente.

Después ponemos el campo nombre, tipo VARCHAR con 50 caracteres de límite y NOT NULL.

Después el apellido, también VARCHAR (50). Después el email, VARCHAR(100) y NOT NULL. Y el teléfono, tipo VARCHAR(20)

* + USE agenda; (pulso enter)
  + CREATE TABLE amigos(

id\_amigo SMALLINT UNSIGNED NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(50) NOT NULL,

apellido VARCHAR(50),

email VARCHAR(100) NOT NULL,

teléfono VARCHAR(20),

PRIMARY KEY (id\_amigo)

);

* Para comprobar que la tabla anterior se ha creado, sin salir de la base de datos escribo:
  + SHOW TABLES;
* Para ver cómo está definida una tabla llamada amigos en una base de datos (vale DESCRIBE y vale DESC):
  + DESCRIBE amigos;
  + DESC amigos;
* Para ejecutar una instrucción desde un archivo con formato .sql (que puede crearse desde el bloc de notas), se pone en la línea de comando SOURCE, luego la ruta y el nombre del archivo con las instrucciones que se quieren ejecutar:
  + SOURCE C:/usuarios/documentos/archivossql/pruebaagenda.sql;
* Añadir en la tabla contactos una columna llamada email al final:
  + ALTER TABLE contactos ADD email VARCHAR (60);
* Añadir en una tabla llamada contactos una columna email detrás de la columna apellido:
  + ALTER TABLE contactos ADD email VARCHAR(60) AFTER apellido;
* Añadir en la tabla contactos una columna llamada email y que sea la primera columna de la tabla:
  + ALTER TABLE contactos ADD email VARCHAR(60) FIRST;
* Cambiar el nombre de un campo que se llama nombre y quiero que se llame apellido:
  + ALTER TABLE contactos CHANGE COLUMN nombre apellido VARCHAR (40);
* Para cambiar tipos de datos y restricciones, se usa ALTER TABLE nombretabla MODIFY COLUMN y simplemente se pone el nombre del campo con sus nuevas propiedades, realmente lo que se hace es redefinirla, así que hay que ponerlas todas como se quieren. Por ejemplo, para cambiar el número de caracteres del campo apellido:
  + ALTER TABLE contactos MODIFY COLUMN apellido VARCHAR (100);
* Para añadir un registro a la tabla contactos, añadiendo solo algunos campos:
  + INSERT INTO contactos (id, nombre, apellido, email, telefono) VALUES (014, ‘Paco’, ‘López’, ‘pacolopez@tratlamdkad.com’, 666666666)
* Para añadir un registro a la tabla contactos completando todos los campos, puede omitirse el primer paréntesis con los campos.
  + INSERT INTO contactos VALUES (014, ‘Paco’, ‘López’, 28, ‘pacolopez@tratlamdkad.com’, 666666666)
* Mostrar el contenido de una tabla. Se usa \* para mostrar todos los registros:
  + SELECT \* FROM contactos;
* Mostrar todo el contenido de una tabla ordenado por puntuación (la tabla tendrá un campo llamado puntuación):
  + SELECT \* FROM socios ORDER BY puntuacion;
* Mostrar los registros de una tabla con puntuación mayor a 15:
  + SELECT \* FROM socios WHERE puntuación >15;
* Mostrar el nombre y el apellido de los registros de una tabla cuyo nombre empiece por una letra igual o anterior a la n en el abecedario (hay que poner las comillas a la ‘n’):
  + SELECT nombre, apellido FROM socios WHERE apellido<’n’;
* En la tabla “productos” mostrar todos los campos de los productos (registros) de la marca seagate y cuyo precio sea superior a 50:
  + SELECT \* FROM productos WHERE marca=”seagate” and precio>50;
* Introducir varios registros al mismo tiempo.
  + INSERT INTO socios (nombre, apellido, email, telefono) VALUES

(‘Ernesto’, ‘Caracesto’, ‘caracesto3000@iefiejeieie.com’, 599459359944),

(‘Ernesta’, ‘Caracesta’, ‘iediwdjiwdw@iefiejf.com, 959349394934),

(‘Mónica’, ‘Pérez’, ‘iwiedjiwd@iefieieff.com, 94593959354);

* Mostrar una tabla pero no entera sino solo los 6 primeros registros:
  + SELECT \* FROM productos LIMIT 6;
* Mostrar una tabla pero no entera sino 7 registros y no desde el principio sino a partir del cuarto (mostraría del 5 al 11):
  + SELECT \* FROM productos LIMIT 4, 7;
* Tengo la tabla alumnos2019 en la base de datos alumnos, y la tabla alumnos2020 en la base de datos alumnos2. Tienen la misma estructura (mismos campos). Quiero meter todos los registros de la tabla alumnos2020 en la tabla alumnos2019:
  + INSERT INTO alumnos.alumnos2019 SELECT \* FROM alumnos2.alumnos2020;
* Mostrar nombre y apellido de los registros de la tabla alumns cuya edad sea mayor de 20 y menor de 27. Se puede hacer con los operadores < y >, y se puede hacer con between.
  + SELECT nombre, apellido FROM alumns where edad <27 and edad > 20;
  + SELECT nombre, apellido FROM alumns WHERE edad BETWEEN 21 AND 26;
* Para abreviar la escritura puede usarse IN. Mostrar nombre marca, unidades y precio de los productos (registros) de las marcas “toshiba”, “Samsung” y “western” en la tabla productos.
  + SELECT nombre, marca, unidades, precio FROM productos WHERE marca=”Toshiba” OR marca=”Samsung” OR marca=”western”;
  + SELECT nombre, marca, unidades, precio FROM productos WHERE marca IN (“Toshiba”, “Samsung”, “western”)
* Mostrar nombre, marca, unidades y precio de todos los registros que en su campo nombre empiecen por “Disco interno”, en la tabla productos:
  + SELECT nombre, marca, unidades, precio FROM productos WHERE nombre LIKE “Disco interno%”;
* Mostrar nombre, marca, unidades y precio de todos los registros que en su campo nombre terminen por “2 Tb”, en la tabla productos:
  + SELECT nombre, marca, unidades, precio FROM productos WHERE nombre LIKE “%2 Tb”;
* Mostrar nombre, marca, unidades y precio de todos los registros que en su campo nombre contengan en cualquier posición dentro del campo el texto “HDD” en la tabla productos:
  + SELECT nombre, marca, unidades, precio FROM productos WHERE nombre LIKE “%HDD%”;
* Mostrar nombre, marca, unidades y precio del registro que en el campo nombre contenga el texto 1TB o 1 TB (con espacio entre el número y el TB o sin espacio, porque puede que algunos estén puestos con espacio y algunos sin él):
  + SELECT nombre, marca, unidades, precio FROM productos WHERE nombre LIKE “%1%TB%”;
* Mostrar nombre, marca, unidades y precio de los registros cuya marca sea una palabra que termina por “estern” y que sabemos que tiene otra letra antes pero no recordamos cuál es esa letra, pero sabemos que solo es una:
  + SELECT nombre, marca, unidades, precio FROM productos WHERE marca LIKE “\_estern”;
* Mostrar qué marcas distintas existen dentro del campo marcas en la tabla productos (da todas resultados del campo marca sin repetir):
  + SELECT DISTINCT marca FROM productos;
* Mostrar las distintas combinaciones de marcas y unidades en la table productos (devuelve los resultados en los que son distintos o la marca o el número de unidades). Si de una marca hay 7 productos distintos y quedan 5 unidades de cada uno de ellos, solo dará un resultado, pero si de uno de ellos quedan solo 4 ó 3 unidades, dará un resultado por cada pareja marca-unidades distinta:
  + SELECT DISTINCT marca, unidades FROM productos;
* Mostrar nombre, marca, unidades y precio de los registros de la table productos ordenados por precio (si no se especifica nada, es orden ascendente es lo mismo que ponerle ASC):
  + SELECT nombre, marca, unidades, precio FROM productos ORDER BY precio;
  + SELECT nombre, marca, unidades, precio FROM productos ORDER BY precio ASC;
* Mostrar nombre, marca, unidades y precio de los registros de la table productos ordenados por precio en orden descendente:
  + SELECT nombre, marca, unidades, precio FROM productos ORDER BY precio DESC;
* Mostrar los registros de la tabla productos cuyo número de unidades sea mayor que 0 y ordenados por precio ascendente:
  + SELECT \* FROM productos WHERE unidades>0 ORDER BY precio;
* Mostrar los registros de la tabla productos cuyo número de unidades sea mayor que 0 y ordenados por marca primero (en orden alfabético ascendente) y por precio ascendente dentro de cada marca:
  + SELECT \* FROM productos WHERE unidades>0 ORDER BY marca, precio;
* Mostrar los alumnos de entre 25 y 30 años junto con sus tutores (otro campo en la misma tabla) de la tabla alumnos, haciendo que la columna alumnos se muestre con el nombre “chavales” y la columna tutores se muestre como “paisanos”:
  + SELECT nombre AS chaval, tutor AS paisano FROM alumnos WHERE edad BETWEEN 24 AND 31;
* Si tengo una tabla llamada productos, con discos duros y resulta que la marca western cambia su nombre y pasa a llamarse westgate (por ejemplo), para cambiar la marca en todos los discos duros de la marca western por westgate:
  + UPDATE productos SET marca=”westgate” WHERE marca=”western”;
* Si un disco duro de 1 tb de la marca Seagate cambia de precio y quiero cambiarle el precio, OJO, lo cambio con un campo único, ya que si pongo como criterio discos de 1 tb puede haber más de uno y me los cambia todos. Así que se suele usar el ID o lo que sea la clave primaria:
  + UPDATE productos SET precio = 38,50 WHERE id=98;
* Para sumar un número a un valor guardado. Por ejemplo, en la tabla productos tengo (en el campo unidades) 26 unidades del producto que tiene como identificador (clave primaria) el valor “DE27” y recibo un lote con 100 más. (Podría hacer yo la suma y poner 126 como nuevo valor, pero mysql hace la suma si se lo digo). Para sumar los 100 nuevos a los que ya tenía:
  + UPDATE productos SET unidades = unidades+100 WHERE id=”DE27”.
* Tenemos una tabla productos, la clave primaria es el id, tiene otro campo que es precio y otro campo que es unidades (número de unidades existentes de ese producto). Recibo 20 unidades nuevas y además quiero rebajar el precio a la mitad. Puedo meter todo junto:
  + UPDATE productos SET unidades = unidades + 20, precio = precio\*0,5 WHERE id=”DE23”;
* Para cambiar por ejemplo el número de unidades en el campo unidades, en una tabla llamada productos, y que ese cambio se haga a TODOS los registros, se hace sin poner la cláusula WHERE. Es importante esa cláusula si no se quieren cambiar todos los registros de la tabla, porque te los cargas todos si no la pones y no querías cambiarlos todos:
  + UPDATE productos SET unidades=0;
* Si cuando estoy creando una tabla tengo un campo fecha y quiero que por defecto cada registro tenga como fecha la del momento en que se crea el registro, indico al crear la tabla, al poner el tipo de dato que es la fecha, la cláusula CURRENT TIMESTAMP o también puede ponerse NOW. Hace lo mismo:
  + CREATE TABLE artículos(id articulo INT UNSIGNED AUTO INCREMENT, fecha DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP, …. ;
  + CREATE TABLE artículos(id articulo INT UNSIGNED AUTO INCREMENT, fecha DATETIME NOT NULL DEFAULT NOW, …. ;
* Eliminar la tabla clients:
  + DROP TABLE clients;

BUSCAR:

Trigger