**Universidad de san buenaventura**

**Mario Juan Sebastian Reyes Casas**

**Bitácora SQL 2.0**

**Luis Barreto**

**Bogotá D.C**

**12/03/2020**

**Índice:**

**Create db**

**Drop**

**Backup**

**Create table**

**Drop table**

**Alter table**

**Constraints**

**Not null**

**Unique**

**Primary key**

**Foreign key**

**Check**

**Default**

**Index**

**Auto increment**

**Dates**

**Views**

**Injection**

**Hosting**

**Create db**

create database prueba1;

Query OK, 1 row affected (0.06 sec)

+--------------------+

| Database |

+--------------------+

| cultivos |

| datawharehouse |

| guia |

| information\_schema |

| mysql |

| performance\_schema |

| prueba1 |

| test |

+--------------------+

8 rows in set (0.01 sec)

con este query podemos crear una base de datos cualquiera .

**Drop database**

drop database prueba1;

Query OK, 0 rows affected (0.39 sec)

+--------------------+

| Database |

+--------------------+

| cultivos |

| datawharehouse |

| guia |

| information\_schema |

| mysql |

| performance\_schema |

| test |

+--------------------+

7 rows in set (0.09 sec)

Aplicando este comando podemos borrar toda una base de datos con las tablas que esta tiene adentro .

**Backup**

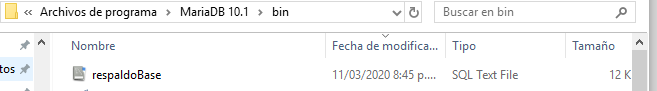
Utilizamos el siguiente query que nos genera :

mysqldump -u root -p cultivos > respaldoBase.sql

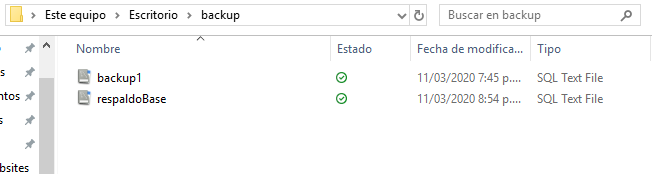
Enter password: \*\*\*\*\*\*\*\*

mysqldump -u root -p cultivos > C:\Users\marse\OneDrive\Escritorio\backup\respaldoBase.sql

Enter password: \*\*\*\*\*\*\*\*



aca podemos observar nuestra copia de seguridad de la base de datos con el primer quey se guarda en el directorio donde hemos ejecutado el comando sin embargo se puede ejecutar en una carpeta aparte como lo es el segundo query su resultado es :



Es importante que si nuestro SGBDD se encuentra en windows y en las carpetas de “archivos de programas” ,abramos la cmd como administrador

**Create table**

create table estudia as select ced , nombre , edad from datawharehouse.estudiantes;

Query OK, 9 rows affected (0.23 sec)

Records: 9 Duplicates: 0 Warnings: 0

MariaDB [pruebadata]> show tables;

+----------------------+

| Tables\_in\_pruebadata |

+----------------------+

| estudia |

+----------------------+

1 row in set (0.00 sec)

El siguiente query nos ayudo a crear una tabla basado en otra tabla que ya existía en una base de datos diferente

create table estudia2 (id\_estu int(11) ,nombre varchar(80),apellido varchar(80));

Query OK, 0 rows affected (0.27 sec)

Acá se crea la tabla estudia2 pero con las columnas que nosotros le hemos designado .

**Drop table**

drop table estudia2;

Query OK, 0 rows affected (0.32 sec)

Este es el resultado que nos devuelve después de haber hecho un drop que nos es mas que borrar la base de datos

**Alter table**

alter table estudia add celular varchar(255);

Query OK, 0 rows affected (0.49 sec)

Records: 0 Duplicates: 0 Warnings: 0

Este query nos permite agregar un campo a la base de datos en este caso le hemos agregado el numero de celular

alter table estudia drop column celular ;

Query OK, 0 rows affected (0.30 sec)

Records: 0 Duplicates: 0 Warnings: 0

Ahora si queremos remover una columna simplemente con el query alter table drop columna podemos quitarlo sin ningún problema

**Constraints**

**Not null**

Para evitar que los campos se llenen de campos nulos utilizamos este query

create table estudiante ( id int(11) not null ,nombre varchar(255) not null );

Query OK, 0 rows affected (0.27 sec)

Con esto evitamos que los campos id y nombre queden nulos y obliga a que sean llenados con algún valor , lo mismo se puede hacer en los quey alter table

alter table estudiante add correo varchar (255) not null ;

Query OK, 0 rows affected (0.41 sec)

Records: 0 Duplicates: 0 Warnings: 0

**Unique**

alter table estudiante add constraint id unique(id) ;

Query OK, 0 rows affected (0.34 sec)

Records: 0 Duplicates: 0 Warnings: 0

Este termino nos ayuda a que ningún termino se nos repita en la columna que estamos especificando , este termino junto con primary key nos ayudan a generar este tipo de cosas .

Así como se agrego un unique constraint también se puede quitar con el comando “**drop index**”

**Primary key**

create table estudiante2 (id int(11) , nombre varchar(255) , primary key (id));

Query OK, 0 rows affected (0.37 sec)

Como se ha dicho anteriormente la primary key no nos permite que un dato se repita asi como lo hace unique , además podemos agregar este index a través del comando alter table

alter table estudiante add primary key (id);

Query OK, 0 rows affected (0.41 sec)

Records: 0 Duplicates: 0 Warnings: 0

**Foreign key**

Para este ejercicio tenemos dos tablas estudiante y materia , para agregar la llave foránea hay que saber quien se lleva quien , recordemos la relaciones 1:M , 1:1 ,M:M , para no entrar en tanto detalle las que son muchos a muchos y las que son uno a uno generan una nueva tabla , mientras las relaciones de 1 a muchos la de muchos se traen la llave foránea de uno

create table materia (idMat int (11) , nombre varchar(255),idest int (11), primary key (idMat) , constraint estude foreign key (idest) references estudiante2(id));

Query OK, 0 rows affected (0.27 sec)

Acá se creo la tabla materia que posteriormente traerá el id del estudiante en la llave foránea ,la tabla materia se hubiera podido crear sin la llave foránea y se hubiera podido agregar con un “**alter table**”

**Check:**

alter table estudiante add celular int (12) check (celular >= 3000000000 and celular <= 4000000000);

Query OK, 0 rows affected (0.45 sec)

Records: 0 Duplicates: 0 Warnings: 0

Este query nos permite establecer un limite que puede tener un registro , es decir los datos no pueden pasar de cierto valor ,esto se puede realizar al inicio de la creación de la tabla ,como ejemplo se ha tomado cuando la tabla ya se ha creado, también cuenta con la propiedad de borrar el check

**Default:**

alter table estudiante alter ciudad set default 'Bogotá';

Query OK, 0 rows affected (0.10 sec)

Records: 0 Duplicates: 0 Warnings: 0

Para crear un campo default hemos hecho dos alter uno para crear el campo ciudad y el otro para generar por default la ciudad de Bogotá ,como tal default solo nos genera el valor que quedara por defecto en caso de que no se llene con algo el campo o que quede null .

También hereda los alter drop default en caso de querer quitar este campo por default

**Index**

create index id\_nombre on estudiante (nombre);

Query OK, 0 rows affected (0.28 sec)

Records: 0 Duplicates: 0 Warnings: 0

show index from pruebadata.estudiante;

+------------+------------+-----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+

| Table | Non\_unique | Key\_name | Seq\_in\_index | Column\_name | Collation | Cardinality | Sub\_part | Packed | Null | Index\_type | Comment | Index\_comment |

+------------+------------+-----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+

| estudiante | 0 | PRIMARY | 1 | id | A | 3 | NULL | NULL | | BTREE | | |

| estudiante | 0 | id | 1 | id | A | 3 | NULL | NULL | | BTREE | | |

| estudiante | 1 | id\_nombre | 1 | nombre | A | 3 | NULL | NULL | | BTREE | | |

+------------+------------+-----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+

3 rows in set (0.10 sec)

Este query ayuda a organizar la información de una manera más rápida , asi que simplemente usamos el comando de arriba y obtenemos con un show

**Auto incremental:**

alter table estudiante modify column id int (11) not null auto\_increment;

Query OK, 4 rows affected (0.79 sec)

Records: 4 Duplicates: 0 Warnings: 0

Con este pequeño query lo que se ha hecho es asignarle un valor auto incremental a la columna id , entonces si alguien no ingresa el valor del id este automáticamente se generara

**VIEWS**

**create** **view** nom **AS** **select** nombre , ciudad **from** estudiante;

show tables;

+----------------------+

| Tables\_in\_pruebadata |

+----------------------+

| estudiante |

| estudiante2 |

| materia |

| nom |

+----------------------+

4 rows in set (0.10 sec)

Las vistas son una especie de colección de consultas ,en ellas podemos almacenar nuestras consultas para que estas no se pierdan

Si se quiere borrar una vista simplemente se usa el comando “**DROP**”

**SQL Injection**

Es un ataque que se hace para sacar de sentencias sql información delicada

**select \* from estudiante where nombre='jose' or 1=1;**

**+----+---------+----------+-----------+--------+**

**| id | nombre | correo | celular | ciudad |**

**+----+---------+----------+-----------+--------+**

**| 1 | juan | oasmdasl | 3165542 | NULL |**

**| 2 | josue | mardeñ | 300002343 | Bogotá |**

**| 3 | mar | modelia | 3455666 | Bogotá |**

**| 4 | josel | fsdfs | 3455666 | Bogotá |**

**| 5 | bjhbjhj | jhbjhbhj | 344343 | Bogotá |**

**+----+---------+----------+-----------+--------+**

**5 rows in set (0.01 sec)**

Aca se puede observar como se selecciona todo si nombre es jose o si por el contrario 1 es igual a 1 y como en el OR la condición se cumple siempre y cuando una sola se cumpla ,devuelve todos los datos