Bases de datos

ELIANA YINETH
LOZANO TRIANA
eylozano@sena.edu.co

ACTUALIZAR Y ELIMINAR REGISTROS

Estimados aprendices, ya sabemos insertar y consultar registros de nuestras tablas, ahora nos corresponde actualizar y eliminar todos aquellos registros que previamente hemos trabajado.

Primero que todo, por favor ingresen a este link y actualicen sus datos por favor

https://docs.google.com/spreadsheets/d/10xBBKTDBVOXIx0ndr9ymsYKs7QU0ahTy/edit?usp=sharing&ouid=106229533260816064003&rtpof=true&sd=true

Organicemos la base en la que vamos a trabajar.

--------- Base de datos de demostración

POR FAVOR CREAR UNA BASE CON LAS SIGUIENTES CARACTERÍSTICAS PARA QUE REALICEN CADA UNO DE LOS PASOS Y CASOS DE ESTA GUÍA, CARGAR EL ARCHIVO .SQL Y EL WORD CON LOS PANTALLAZOS AL REPO COMO ACTU.ELIM_SQL EN UN ARCHIVO ZIP

1. **CREATE DATABASE** CADPH;
2. **USE** CADPH;
3. **CREATE TABLE** Titulada(

   id INT (20) UNIQUE PRIMARY KEY,

   nombre_apellido VARCHAR (50) UNIQUE NOT NULL,

   correo VARCHAR(50) UNIQUE NOT NULL,

   edad INT UNSIGNED NOT NULL,

   direccion VARCHAR(20) NOT NULL,

   ciudad VARCHAR(20) NOT NULL,

   estado ENUM('Activo', 'Inactivo') DEFAULT 'Inactivo',

   formación ENUM('Técnico', 'Tecnólogo') DEFAULT 'Tecnólogo',

   creado TIMESTAMP DEFAULT CURRENT_TIMESTAMP

   );
4. Realizar inserción de 20 registros.

   INSERT INTO Titulada (id, nombre_apellido, correo, edad, direccion, ciudad, estado, formación)
   VALUES …

--------- Base de datos de demostración

Y esto queda así:

```
mysql> desc Titulada;
+-----------------+-----------------------------+------+-----+-------------------+-------------------+
| Field           | Type                        | Null | Key | Default           | Extra             |
+-----------------+-----------------------------+------+-----+-------------------+-------------------+
| id              | int                         | NO   | PRI | NULL              |                   |
| nombre_apellido | varchar(50)                 | NO   | UNI | NULL              |                   |
| correo          | varchar(50)                 | NO   | UNI | NULL              |                   |
| edad            | int unsigned                | NO   |     | NULL              |                   |
| direccion       | varchar(20)                 | NO   |     | NULL              |                   |
| ciudad          | varchar(20)                 | NO   |     | NULL              |                   |
| estado          | enum('Activo','Inactivo')   | YES  |     | Inactivo          |                   |
| formación       | enum('Técnico','Tecnólogo') | YES  |     | Tecnólogo         |                   |
| creado          | timestamp                   | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED |
+-----------------+-----------------------------+------+-----+-------------------+-------------------+
9 rows in set (0.01 sec)

mysql> select * from titulada;
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
| id        | nombre_apellido | correo         | edad | direccion | ciudad | estado   | formación | creado              |
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
| 107587125 | Diego Rojas     | test4@test.com |   25 | Calle 10  | Bogotá | Inactivo | Técnico   | 2023-06-13 14:53:55 |
| 107786124 | Ana Torres      | test1@test.com |   20 | Calle 13  | Neiva  | Activo   | Técnico   | 2023-06-13 14:53:55 |
| 107786125 | Julian Castro   | test2@test.com |   30 | Calle 11  | Garzón | Activo   | Tecnólogo | 2023-06-13 14:53:55 |
| 107787125 | Camilo Gutierrez | test3@test.com |   25 | Calle 15  | Cali   | Activo   | Tecnólogo | 2023-06-13 14:53:55 |
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
4 rows in set (0.00 sec)
```

UPDATE

La palabra reservada Update nos permite realizar una actualización a las rows de nuestra tabla, aquí la sintaxis.

UPDATE <Nombre_de_la_tabla> **SET** <Columna_a_modificar>= ‘Lo que vamos a actualizar;

Sin embargo, el UPDATE nos permite ingresar una condición o regla, dado que como está la sintaxis arriba, nos actualizaría todos los registros contenidos para esa columna, siendo que, solo queremos actualizar un ID en específico para esa columna.

Entonces, esto quedaría así:

UPDATE <Nombre_de_la_tabla> **SET** <Columna_a_modificar>= ‘Lo que vamos a actualizar’ **WHERE** <identificador de registro> = <valor>;

Si quisiera actualizar la columna **ESTADO**, del registro con **ID 107786124.** Quedaría así:

UPDATE Titulada **SET** estado = ‘Inactivo’ **WHERE** id = 107786124;

Antes:

```
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
| id        | nombre_apellido | correo         | edad | direccion | ciudad | estado   | formación | creado              |
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
| 107587125 | Diego Rojas     | test4@test.com |   25 | Calle 10  | Bogotá | Inactivo | Técnico   | 2023-06-13 14:53:55 |
| 107786124 | Ana Torres      | test1@test.com |   20 | Calle 13  | Neiva  | Activo   | Técnico   | 2023-06-13 14:53:55 |
| 107786125 | Julian Castro   | test2@test.com |   30 | Calle 11  | Garzón | Activo   | Tecnólogo | 2023-06-13 14:53:55 |
| 107787125 | Camilo Gutierrez | test3@test.com |   25 | Calle 15  | Cali   | Activo   | Tecnólogo | 2023-06-13 14:53:55 |
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
```

Ahora:

```
mysql> SELECT * FROM titulada;
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
| id        | nombre_apellido | correo         | edad | direccion | ciudad | estado   | formación | creado              |
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
| 107587125 | Diego Rojas     | test4@test.com |   25 | Calle 10  | Bogotá | Inactivo | Técnico   | 2023-06-13 14:53:55 |
| 107786124 | Ana Torres      | test1@test.com |   20 | Calle 13  | Neiva  | Inactivo | Técnico   | 2023-06-13 14:53:55 |
| 107786125 | Julian Castro   | test2@test.com |   30 | Calle 11  | Garzón | Activo   | Tecnólogo | 2023-06-13 14:53:55 |
| 107787125 | Camilo Gutierrez | test3@test.com |   25 | Calle 15  | Cali   | Activo   | Tecnólogo | 2023-06-13 14:53:55 |
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
4 rows in set (0.00 sec)
```

Ejercicio 1

Actualizar las 8 columnas, en por lo menos 1 valor, para los 20 registros.

Bases de datos

ELIANA YINETH
LOZANO TRIANA
eylozano@sena.edu.co

DELETE

La palabra reservada Delete, nos permite eliminar las rows o registros de nuestra tabla, aquí la sintaxis.

DELETE FROM <Nombre de la tabla>;

Sin embargo, el DELETE nos permite ingresar una condición o regla, dado que como está la sintaxis arriba, nos eliminaría todos los registros contenidos para esa tabla, siendo que, solo queremos un registro en específico.

Entonces, esto quedaría así:

DELETE FROM <Nombre de la tabla> **WHERE** <identificador de registro> = <valor>;

Si quisiera eliminar el registro con **ID 107786124.** Quedaría así:

DELETE FROM titulada **WHERE** id = 107786124;

```
mysql> DELETE  FROM titulada WHERE id = 107786124;
Query OK, 1 row affected (0.02 sec)
```

```
mysql> SELECT * FROM titulada;
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
| id        | nombre_apellido | correo         | edad | direccion | ciudad | estado   | formación | creado              |
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
| 107587125 | Diego Rojas     | test4@test.com |   25 | Calle 10  | Bogotá | Inactivo | Técnico   | 2023-06-13 14:53:55 |
| 107786125 | Julian Castro   | test2@test.com |   30 | Calle 11  | Garzón | Activo   | Tecnólogo | 2023-06-13 14:53:55 |
| 107787125 | Camilo Gutierrez | test3@test.com |   25 | Calle 15  | Cali   | Activo   | Tecnólogo | 2023-06-13 14:53:55 |
+-----------+-----------------+----------------+------+-----------+--------+----------+-----------+---------------------+
3 rows in set (0.00 sec)
```

TRUNCATE TABLE

En esencia, el **TRUNCATE TABLE**, permite eliminar registros de la tabla igual que el **DELETE** en su forma básica, es decir, sin el **WHERE** o regla. Sin embargo, el Truncate sirve para mucho más que eliminar, nos permite realizar una restauración de la meta información de la tabla, esto que quiere decir, que si tenemos auto incrementales estos iniciarán nuevamente desde 1, actúa como un refresh para la tabla.

```
mysql> TRUNCATE TABLE titulada;
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> SELECT * FROM titulada;
Empty set (0.00 sec)
```

Ahora eliminemos la base de datos trabajada:

```
mysql> SHOW databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
6 rows in set (0.00 sec)
```

Bases de datos

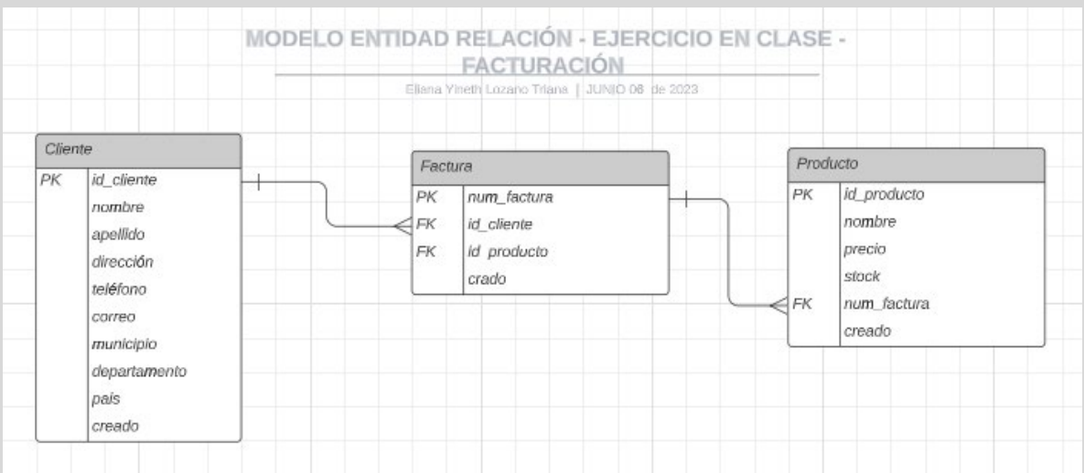
ELIANA YINETH
LOZANO TRIANA
eylozano@sena.edu.co

RELACIONES

Cuando se está realizando el maquetado de nuestras tablas, utilizamos principalmente 3 tipos de relaciones:

- Uno a uno.
- Uno a muchos.
- Muchos a muchos.

Crearemos la siguiente base con sus correspondientes tablas, la relación más normal de encontrar o utilizar es de ***uno a muchos***, para esto aremos uso de llaves foráneas, que no es más que una columna que almacena una llave primaria de otra tabla o tabla externa.



Un cliente → Puede tener muchas facturas.

Una factura → Puede tener muchos productos.

Es decir que para este ejercicio aremos uso de la relación uno a muchos.

Bases de datos

ELIANA YINETH
LOZANO TRIANA
eylozano@sena.edu.co

```sql
CREATE DATABASE FACTURACIÓN;
USE FACTURACIÓN;

CREATE TABLE cliente(
    id_cliente VARCHAR (30) UNIQUE PRIMARY KEY,
    nombre VARCHAR (25) UNIQUE NOT NULL,
    apellido VARCHAR (25) UNIQUE NOT NULL,
    direccion VARCHAR(20) NOT NULL,
    telefono VARCHAR(20) NOT NULL,
    correo VARCHAR(50) UNIQUE NOT NULL,
    municipio VARCHAR(20),
    departamento VARCHAR(20),
    pais VARCHAR (20) CHECK (pais='Colombia'), -- CHECK permite limitar a una respuesta
    creado TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE factura(
    num_factura VARCHAR (20) UNIQUE PRIMARY KEY,
    id_cliente VARCHAR (30),
    id_producto VARCHAR (30),
    creado TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente),
    FOREIGN KEY (id_producto) REFERENCES productos(id_producto)
);

CREATE TABLE productos(
    id_producto VARCHAR (30) UNIQUE PRIMARY KEY,
    nombre VARCHAR (25) NOT NULL,
    precio INT (25) NOT NULL,
    stock INT (25) NOT NULL,
    num_factura VARCHAR (20),
    creado TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (num_factura) REFERENCES factura(num_factura)
);
```

Hasta acá todo bien?, pero.. se estarán preguntando por que en la tabla “Factura”, está generando un error en la última llave foránea, correcto?

Básicamente es por que no podemos declarar una llave foránea que toma como referencia una tabla que no hemos creado, como es el caso de productos, y si crean primero productos pues tampoco lo podrán hacer dado que la tabla factura no estaría creada.

EJERCICIO 2

Entonces investigue como asignar esta clave foránea faltante, haciendo uso de ALTER TABLE (Mediante la instrucción ALTER TABLE se puede modificar una tabla existente de varias formas) o de otro método para esta alteración o modificación.

EJERCICIO 3

Posterior inserte 10 productos, 5 clientes y 10 facturas.

Bases de datos

ELIANA YINETH
LOZANO TRIANA
eylozano@sena.edu.co

Funciones de agregación

Las funciones de agregación en SQL nos permiten efectuar operaciones sobre un conjunto de resultados, pero devolviendo un único valor agregado para todos ellos. Es decir, nos permiten obtener medias, máximos, etc... sobre un conjunto de valores.

Las funciones de agregación básicas que soportan todos los gestores de datos son las siguientes:

- COUNT: devuelve el número total de filas seleccionadas por la consulta.
- MIN: devuelve el valor mínimo del campo que especifiquemos.
- MAX: devuelve el valor máximo del campo que especifiquemos.
- SUM: suma los valores del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.
- AVG: devuelve el valor promedio del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.

Cabe aclarar que éstas se realizan con valores numéricos

Función SUM() → Devuelve la suma de todos los valores de una columna.

```
SELECT SUM (columna) FROM nombre_tabla;
```

Función AVG() → Devuelve la media o el promedio de una columna.

```
SELECT AVG (columna) FROM nombre_tabla;
```

Función MIN() → Devuelve el valor mínimo de una columna.

```
SELECT MIN (columna) FROM nombre_tabla;
```

Función MAX() → Devuelve el valor máximo de una columna.

```
SELECT MAX (columna) FROM nombre_tabla;
```

Función COUNT() → Devuelve el número de filas.

```
SELECT COUNT(columna) FROM nombre_tabla;
```

EJERCICIO 4

Realice en base a cada una de estas funciones de agregación por lo menos una consulta.