MÓDULO 6:

CRUD.

Clave primaria.

SQL segunda parte.

Clave foránea.

1000 PROGRAMADORES

>CRUD

El concepto **CRUD** está estrechamente vinculado a la gestión de datos digitales. **CRUD** hace referencia a un acrónimo en el que se reúnen las primeras letras de las cuatro operaciones fundamentales de aplicaciones persistentes en sistemas de bases de datos:

Create (Crear registros)

Read (Leer registros)

Update (Actualizar registros)

Delete (Borrar registros)

>CRUD

En pocas palabras, **CRUD** resume las funciones requeridas por un usuario para crear y gestionar datos. Varios procesos de gestión de datos están basados en **CRUD**, en los que dichas operaciones están específicamente adaptadas a los requisitos del sistema y de usuario, ya sea para la gestión de bases de datos o para el uso de aplicaciones

CRUD-Operation	SQL	RESTful HTTP
Create	INSERT	POST, PUT
Read	SELECT	GET, HEAD
Update	UPDATE	PUT, PATCH
Delete	DELETE	DELETE

* 2022 / 3° Cohorte.

>Crear una tabla con campos

La sintaxis de como crear una tabla es la siguiente:

CREATE TABLE *Nombre_Tabla* (
‘Nombre_Columna1’ **Tipo_de_Dato** (longitud),
‘Nombre_Columna2’ **Tipo_de_Dato** (longitud),
‘Nombre_Columna3’ **Tipo_de_Dato** (longitud),
….);

>Crear una tabla con campos

Los parámetros «Nombre_Columna» especifican los nombres de las columnas que integran las tablas.

Los parámetros «Tipo_de_Dato» especifican que tipo de datos admitirá esa columna (ej. **varchar**, **integer**, **decimal**, etc.).

El parámetro «Longitud» especifica la longitud máxima de caracteres que admitirá la columna de la tabla.

>Crear una tabla con campos

Cabe aclarar que hay varias propiedades que pueden ser aplicadas a las columnas de la tabla, por ejemplo una de las más comunes es **PRIMARY KEY**, la cual nos permite indicar qué columna será llave primaria, también podemos establecer que alguna columna no acepte valores nulos, a través de la propiedad **NOT NULL**.

>Ejemplo: Crear una tabla con campos

```
CREATE TABLE articulos (
        codigo INT PRIMARY KEY NOT NULL,
        descripcion TEXT(100),
        precio REAL
        );
```

>Ejemplo: Crear una tabla con campos

En el código anterior estamos creando la tabla “articulos” que tiene 3 columnas, podemos darnos cuenta de que una de las novedades es en la columna **código**, ya que tenemos la propiedad **PRIMARY KEY**, la cual indica que es la clave primaria que identifica de manera única cada registro/fila de la tabla, así mismo con la propiedad **not null** estamos indicando que esa columna es indispensable, es decir que siempre debe ser ingresado un **código**.

>Ejemplo: Crear una tabla con campos

Cabe aclarar que la propiedad **not null** puede ser implementada en otras columnas que consideremos que son indispensables, por ejemplo **descripcion** es fundamental que sea ingresado, en ese caso deberíamos agregarle la propiedad **not null** para garantizar que el nombre siempre será ingresado.

Para efectos de este ejemplo no ha sido aplicado; sin embargo, cuando estamos creando unas Base de Datos con una buena estructura, debemos tomar en cuenta todos esos aspectos.

>Clave primaria

La clave primaria, **PRIMARY KEY**, identifica de manera única cada fila de una tabla.

La columna definida como clave primaria (**PRIMARY KEY**) debe ser **UNIQUE** (valor único) y **NOT NULL** (no puede contener valores nulos).

Cada tabla solo puede tener una clave primaria (**PRIMARY KEY**).

Una llave primaria o **PRIMARY KEY** es una columna o un grupo de columnas que identifica de forma exclusiva cada fila de una tabla. Puede crear una llave primaria para una tabla utilizando la restricción **PRIMARY KEY.**

>Clave primaria

Cada tabla puede contener solo una clave primaria. Todas las columnas que participan en la llave primaria **deben** definirse como **NOT NULL**

>Clave primaria

SINTAXIS PARA CREAR LA CLAVE AL MOMENTO DE DEFINIR EL CAMPO QUE SERÁ CLAVE PRIMARIA

```
CREATE TABLE table_name (
    pk_columna tipodeDato PRIMARY KEY NOT NULL,
    ...
    ...
    );
```

>Clave primaria

Ejemplo **PRIMARY KEY** o clave primaria en MySQL

La clave primaria se puede agregar al momento de crear el campo que será la clave primaria.

```
CREATE TABLE articulos (
    codigo INT PRIMARY KEY NOT NULL,
    descripcion TEXT,
    precio REAL
    );
```

módulo 6.

1000 PROGRAMADORES

* 2022 / 3° Cohorte.

>Clave primaria autoincremental

El mismo sitio de **SQLite3** dice que no recomienda el autoincremento o las columnas incrementales. Aunque algunas veces es usado para ir incrementando un valor entero y tomarlo como id. Es como la columna autoincremental de **MySQL**.

Una columna numérica de una tabla que es clave **primary** le podemos asignar que sea autoincrementable lo que significa que su valor se incrementa automáticamente al momento de insertar un nuevo registro, por lo cual el valor del id o clave primaria no debe ser asignado al momento se inserta un nuevo registro.

>Clave primaria autoincremental

ID	Nombre	Apellido	Calle	Ciudad
1	Juan	Uno	Calle 274	Veraguas
2	Pedro	Dos	Calle 004	Chiriqui
3	Miguel	Tres	Calle 174	Darien
4	Daniel	Cuatro	Calle 284	Colon
5	Ramiro	Cinco	Calle 277	Panama

Podemos observar aquí como los IDs van tomando valores en orden creciente.

>Ejemplo - PK autoincremental

Ejemplo de como hacer que mi campo de clave primaria sea auto incremental:

```
CREATE TABLE articulos(
    codigo INT(4) PRIMARY KEY  NOT NULL AUTOINCREMENT,
    descripcion TEXT,
    precio REAL
    );
```

>Consultas a una tabla SELECT

SELECT columnas **FROM** nombre_Tabla;

En donde:
SELECT: Es el comando que se utiliza para obtener registros de las tablas.
columnas: Hace referencia a los nombres de las columnas de las cuales queremos consultar registros.
FROM: Es el comando para especificar la tabla de la cual vamos a consultar los datos.
Nombre_Tabla: Hace referencia al nombre de la tabla de la cual queremos consultar los registros.

>Consultas a una tabla SELECT

Todas las columnas. Ejemplo:

- **SELECT** * **FROM** nombreTabla;
- **SELECT** * **FROM** articulos;

***:** Se usa para decir que seleccionamos todas las columnas de la tabla

>Consultas a una tabla SELECT

Algunas columnas:

- **SELECT** columna1, column2 **FROM** articulos;
- **SELECT** descripcion, precio **FROM** articulos;

columna1,columna2: Podemos elegir las columnas que deseemos obtener, **puede poner tantas columnas** como desees.

>Cláusula Where

La cláusula **WHERE** nos permite condicionar las consultas con relación a los registros de una o varias columnas que se especifiquen en la sentencia, en tal sentido todos los registros que entran en el filtro del **where** se mostrarían en los resultados que obtendremos.

La cláusula **where** es utilizada en los casos que no necesitamos que nos muestre todos los registros de una tabla, sino que únicamente los registros que cumplan ciertas condiciones.

>Cláusula Where

Las condiciones conllevan expresiones lógicas que se comprueban con la cláusula **where**. El valor que devuelven las comparaciones es un valor TRUE o FALSE, dependiendo del cumplimiento de la condición especificada.

>Cláusula Where

Podemos hacer uso de cualquier expresión lógica y en ella implementar algún operador de los siguientes:

> “Mayor”,
>= “Mayor o igual”
< “Menor”
<= “Menor o igual”
= “Igual”
<> o != “Distinto”
IS [NOT] NULL “para validar si el valor de una columna no es nulo, es decir, si contiene o no contiene algún registro”
Como así también, los operadores lógicos: **AND** , **OR** y **NOT**

Ejemplo: Clausula Where

```
SELECT *
FROM articulos
WHERE precio = 34;
```

En este caso estaríamos logrando que nos muestre todos los registros que tengan el precio igual a 34.

>Insertar o crear

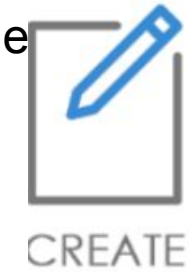
Para agregar datos a las tablas se utiliza la instrucción Insert into, este comando es uno de los más usados en los diferentes gestores de Base de Datos. Para insertar los registros se puede hacer de uno en uno, o agregar varios registros a través de una misma instrucción insert into.
La sintaxis para usar la instrucción insert into en una tabla de MySQL es la siguiente:

```
INSERT INTO “NombreTabla”
    (“PrimeraColumna”, “SegundaColumna”, etc)
VALUES
    (“Dato1”, “Dato2”, etc);
```

>Insertar o crear

Explicación del código:
NombreTabla: Es el nombre de la tabla en la que insertamos registros.
PrimeraColumna, **SegundaColumna**,..: Son las columnas de la tabla en la que vamos a insertar registros.
“**Dato1**”, “**Dato2**” ,..: Son los valores que vamos a guardar en cada columna especificada.
Es importante mencionar que la sintaxis anterior se puede reducir en los casos que vamos agregar datos a todas las columnas, ya que podríamos plantearlo de la forma siguiente:

INSERT INTO “NombreTabla” **VALUES** (“Dato1”, “Dato2”, etc);



>Insertar o crear

Es importante mencionar que la sintaxis anterior se puede reducir en los casos que vamos agregar datos a todas las columnas, ya que podríamos plantearlo de la forma siguiente:

INSERT INTO “NombreTabla” **VALUES** (“Dato1”, “Dato2”, etc);

>Insertar o crear

Cuando aplicamos esta sintaxis, debemos respetar la estructura de la tabla y además después del comando **VALUES** enviar todos los datos para cada columna en el orden correcto, ya que dicha sintaxis indica que vamos a insertar registros a todas las columnas, por lo tanto debemos enviar los datos exactamente como los hemos especificado al momento de crearla.

Existe otra opción de insertar registros y es mencionando columnas específicas, para este caso debemos tomar en cuenta que las columnas que omitimos puedan quedar nulas, es decir que NO le hayamos agregado la propiedad **NOT NULL**.

>Insertar o crear

Ejemplo de insertar un solo registro:

```
INSERT INTO articulos (descripcion, precio)
VALUES ('Manzana',150);
```

>Insertar muchos registros

Si queremos agregar varios registros a través de un mismo insert, basta con agregar una coma en los valores que le enviamos en **VALUES**, y especificar los datos a insertar.

```
INSERT INTO articulo
        (codigo, descripcion, precio)
VALUES
        (30, 'fideos', 200),
        (50, 'jugo', 300),
         (51, 'galletas dulces', 150);
```

>Actualizar registros

Para modificar los datos que contiene actualmente una tabla, se usa la instrucción **UPDATE**, a la que se suele denominar "consulta de actualización". La instrucción **UPDATE** puede modificar uno o varios registros y, por lo general, tiene esta forma.

UPDATE nombreTabla
SET nombreColumna = valor

Para actualizar todos los registros de una tabla, especifique el nombre de la tabla y, después, use la cláusula SET para especificar el campo o los campos que se deben cambiar.

>Actualizar registros - Ejemplo

Ejemplo:

```
UPDATE articulo
SET descripcion = 'Galletas dulces'
```

En la mayoría de los casos, es recomendable que califique la instrucción **UPDATE** con una cláusula **WHERE** para limitar la cantidad de registros que se cambiarán.

```
UPDATE articulos
SET precio = 350
WHERE codigo= 51
```

>Actualizar registros - Ejemplo

Para actualizar más de un campo sólo debemos poner el campo y el valor, por ejemplo:

```
UPDATE articulos
SET descripcion = 'galletas saladas' , precio=100
WHERE id= 51
```

>Eliminar registros de una tabla

Para eliminar los datos que contiene actualmente una tabla, se usa la instrucción **DELETE**, a la que se suele denominar "consulta de eliminación". También se llama "truncar una tabla". La instrucción **DELETE** puede quitar uno o varios registros de una tabla y, en general, tiene esta forma:

sintaxis
DELETE FROM table nombreTabla

>Eliminar registros de una tabla

La instrucción **DELETE** no quita la estructura de la tabla, sino solamente los datos que contiene en ese momento la estructura de la tabla. Para quitar todos los registros de una tabla, use la instrucción **DELETE** y especifique de qué tabla o tablas quiere eliminar todos los registros.

DELETE FROM articulo

>Eliminar registros de una tabla

En la mayoría de los casos, es recomendable que califique la instrucción **DELETE** con una cláusula **WHERE** para limitar la cantidad de registros que se quitarán.

```
DELETE FROM Alumnos
WHERE id = 3
```

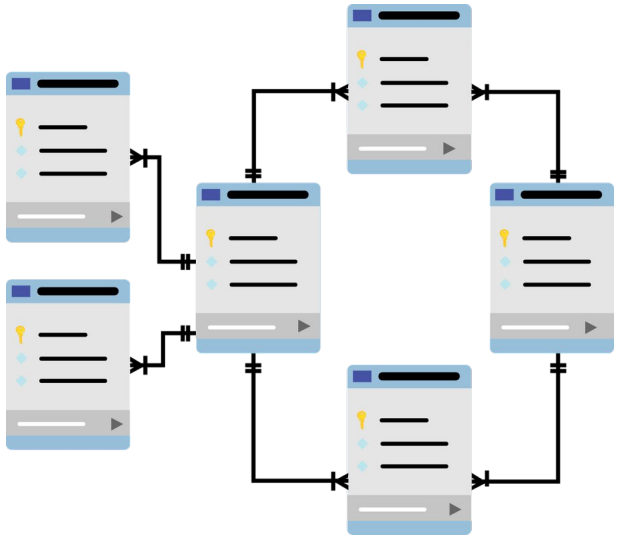
>Relaciones entre Tablas

En el contexto de bases de datos relacionales, una clave foránea o llave foránea o clave ajena es una limitación referencial entre dos tablas. La clave foránea identifica una columna o grupo de columnas en una tabla que se refiere a una columna o grupo de columnas en otra tabla
Una clave foránea se indica al final de la creación de una tabla con:

>Relaciones entre Tablas

FOREIGN KEY(fk) REFERENCES otra_tabla(campo_de_esa_tabla)

Nota: la clave foránea **debe ser** del mismo tipo que la clave primaria a la que se refiere.



módulo 6.

1000 PROGRAMADORES

* 2022 / 3° Cohorte.

>Relaciones entre Tablas

Por ejemplo crear una relación entre la tabla productos y categorías, un producto puede tener una sola categoría pero una categoría puede estar en varios productos, por lo cual la tabla categorías tiene el uno y la tabla de productos tiene la relación de muchos.

```
CREATE TABLE categoria(
        id_cate INTEGER(4) PRIMARY KEY AUTO_INCREMENT,
       descripcion VARCHAR(30)
       );
```

>Relaciones entre Tablas

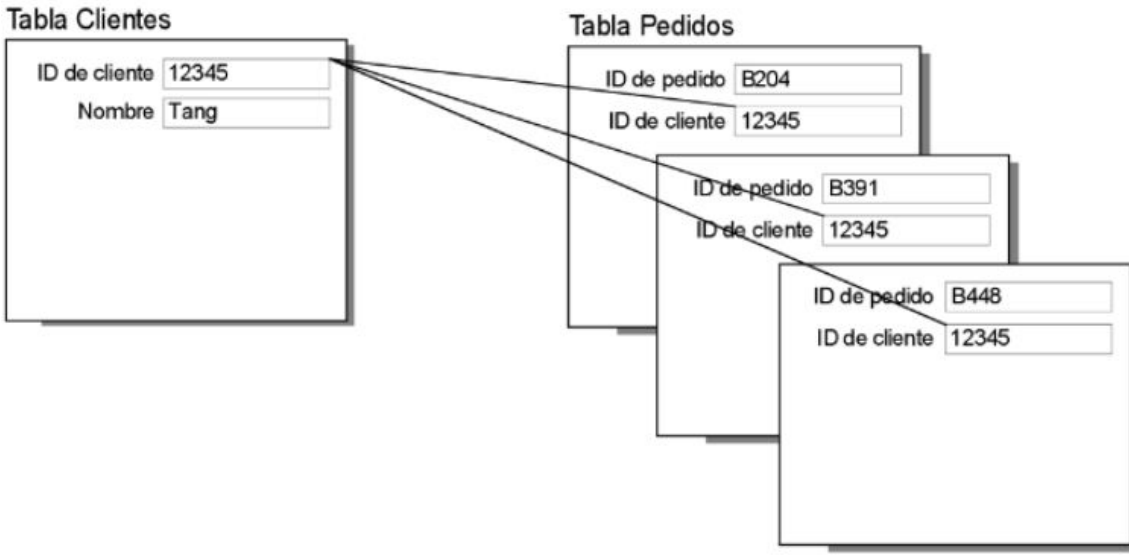
Luego creamos la tabla producto donde crearemos un campo adicional para guardar la clave primaria de la tabla CATEGORIAS

```
CREATE TABLE producto (
id_prod INTEGER PRIMARY KEY AUTO_INCREMENT,
descripcion VARCHAR(30),
precio DOUBLE,
idcategoria INTEGER(4),
FOREIGN KEY(idcategoria) REFERENCES categoria(id_cate)
      );
```

Nota importante la clave primaria de categoría y el campo de clave foránea deben ser del mismo tipo de dato y longitud

>Relaciones de uno a muchos

En una relación de uno a muchos, un registro de una tabla se puede asociar a uno o varios registros de otra tabla.. Por ejemplo, cada cliente puede tener varios pedidos de ventas.



módulo 6.

* 2022 / 3° Cohorte.

>Relaciones de uno a muchos

En este ejemplo, el campo de clave principal de la tabla Clientes, ID de cliente, se ha diseñado para contener valores exclusivos. El campo de clave Foránea de la tabla Pedidos, ID de cliente, se ha diseñado para permitir varias instancias del mismo valor.

Esta relación devuelve registros relacionados cuando el valor del campo ID de cliente de la tabla Pedidos es el mismo que el valor del campo ID de cliente de la tabla Clientes.

web: **http://milprogramadores.unsa.com.ar**

telegram: **https://t.me/milprogramadoressaltenios**

centro de ayuda: **http://ayudamilprogramadores.com/**