CREATE TABLE

En su forma más sencilla, **CREATE TABLE** hará una tabla con todas las columnas que le sean indicadas. A continuación, se presenta un ejemplo, una tabla que admitirá almacenar el nombre de varias personas y las fechas de nacimiento de cada una. Se debe de indicar el nombre de la tabla, y los nombres de las columnas y tipos el tipo de datos de cada una de ella:

Se creará una tabla con el nombre de "**gente**" con dos columnas: "**nombre**" que puede aguantar cadenas de hasta 40 caracteres y "**fecha**" de tipo fecha:

```
mysql> USE prueba;
Database changed
mysql> CREATE TABLE gente (nombre VARCHAR(40), fecha DATE);
Query OK, 0 rows affected (0.53 sec)
mysql>
```

Se puede examinar cuántas tablas y qué nombres tienen en una base de datos, empleando la sentencia **SHOW TABLES**:

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_prueba |
+-----------------+
| gente           |
+-----------------+
1 row in set (0.01 sec)
mysql>
```

a) Valores nulos

Al establecer cada columna, se puede concluir si podrá o no contener valores nulos. Cabe recordar que, aquellas columnas que hacen parte de una clave primaria, no pueden almacenar valores nulos.

Si se define una columna como clave primaria, queda imposibilitada para que pueda contener valores NULL (nulo), pero este caso no es únicamente en que se puede ser interesante imposibilitar la retribución de valores nulos para una columna.

Por defecto, cada columna permite valores nulos (**NULL)**, y para que no se admitan, se usa **NOT NULL**. Por ejemplo:

```
mysql> CREATE TABLE ciudad1 (nombre VARCHAR(20) NOT NULL, poblacion INT NULL);
Query OK, 0 rows affected (0.89 sec)
```

Listar las tablas existentes hasta ahora:

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_prueba |
+-----------------+
| ciudad1         |
| gente           |
+-----------------+
2 rows in set (0.01 sec)

mysql>
```

b) Valores por defecto

En una columna también se puede precisar, opcionalmente, un valor por defecto. El valor por defecto se determinará de forma automática a una columna a la hora que no se especifique un valor determinado al añadir.

Cuando una columna permite tener un valor nulo, y a la hora de insertar no se especifica un valor, **NULL** será usado como valor por defecto. En el anterior ejemplo, el valor por defecto para **poblacion** es **NULL**.

Por ejemplo, si se quiere que 50000 sea el valor que por defecto, se pondrá en la columna **poblacion**, y se creará la tabla como:

```
CREATE TABLE ciudad2 (nombre VARCHAR(20) NOT NULL, poblacion INT NULL DEFAULT 50000);
```

Como la sentencia es larga se digitará por partes así:

1. Primero se escribirá **CREATE TABLE ciudad2 (nombre VARCHAR(20) NOT NULL,**
2. Se presiona la tecla ENTER.
3. Luego se escribe el resto de la sentencia ***poblacion*** **INT NULL DEFAULT 50000);**

Recuerde que MySQL interpreta una sentencia solo cuando encuentra el carácter punto y coma (;). El resultado será:

```
mysql> CREATE TABLE ciudad2 (nombre VARCHAR(20) NOT NULL,
    -> poblacion INT NULL DEFAULT 50000);
```

```
Query OK, 0 rows affected (0.29 sec)
```

c) Claves primarias

Es posible definir una clave primaria sobre una columna, usando la palabra clave **PRIMARY KEY o KEY.**

En cada tabla únicamente puede haber una clave primaria, y la columna sobre la que se concreta una clave primaria no puede poseer valores **NULL**, como ya se hizo mención. Si esto no se especifica de forma explícita, MySQL lo hará de forma automática.

Por ejemplo, si se quiere crear un índice de la tabla de ciudad3 para la columna **nombre**, se creará la tabla así:

```
mysql> CREATE TABLE ciudad3 (nombre VARCHAR(20) NOT NULL PRIMARY KEY,
   -> poblacion INT NULL DEFAULT 50000);
Query OK, 0 rows affected (0. sec)
```

Emplear **NOT NULL PRIMARY KEY**, es lo mismo que **PRIMARY KEY, NOT NULL KEY** o simplemente **KEY**.

Hay una sintaxis distinta para crear claves primarias, que generalmente es preferible, porque es más potente. De hecho, la que se ha explicado es un alias para la forma general, que no permite todas las funcionalidades (como, por ejemplo, crear claves primarias sobre varias columnas).

d) Columnas autoincrementadas

En MySQL existe la posibilidad de crear una columna autoincrementada, sin embargo, esta columna solo puede ser de tipo entero y llave primaria.

Si cuando se inserta una fila se excluye el valor de la columna autoincrementada, o si a un valor nulo se le intenta insertar para esa columna, su valor será calculado automáticamente, tomando el valor mayor de esa columna y sumándole uno, es decir, es secuencial, permitiendo crear una columna con un valor único y secuencias para cada fila de la tabla.

Comúnmente, estas columnas son usadas como claves primarias secuenciales. SQL solo las permite en números enteros, es decir, una columna de otro timo no podrá ser autoincrementable, de tal forma que la composición de clave primaria, que sea entera y autoincrementada, es perfecta para ser usada como clave primaria artificial:

```
mysql > CREATE TABLE ciudad4 (clave INT AUTO_INCREMENT PRIMARY KEY,
   -> nombre VARCHAR(20) NOT NULL,
   -> poblacion INT NULL DEFAULT 50000);
Query OK, 0 rows affected (0.21 sec)


mysql >
```

e) Comentarios

Adicionalmente, al momento de crear la tabla, se puede agregar un comentario a cada columna. Este comentario es usado como información adicional sobre una característica específica de la columna, y va en el apartado de documentación de la base de datos:

```
mysql> CREATE TABLE ciudad5
    -> (clave INT AUTO_INCREMENT PRIMARY KEY COMMENT 'Esto es Clave principal',
    -> nombre VARCHAR(50) NOT NULL,
    -> poblacion INT NULL DEFAULT 50000);
Query OK, 0 rows affected (0.08 sec)
```

f) Descripción de la tabla

SQL tiene una sentencia que permite ver la descripción de una tabla esta sentencia es **DESC** <nombre_tabla>.

```
mysql> DESC ciudad5;
+-----------+----------+------+-----+---------+----------------+
| Field     | Type     | Null | Key | Default | Extra          |
+-----------+----------+------+-----+---------+----------------+
| clave     | int      | NO   | PRI | NULL    | auto_increment |
| nombre    | char(50) | NO   |     | NULL    |                |
| poblacion | int      | YES  |     | 5000    |                |
+-----------+----------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

g) Documentación

MySQL tiene una amplia documentación. Se puede mirar el siguiente enlace para la sintaxis de **crear una tabla** antes de continuar https://dev.mysql.com/doc/refman/8.0/en/create-table.html, luego se podrá extraer una parte para ir estudiando poco a poco.

SQL posee bastante opciones al momento de precisar columnas. También del tipo y el nombre, podemos concretar valores por defecto, admitir o no, que contengan valores tipo NULL, crear una clave primaria, indexar y muchas más. La sintaxis para definir las columnas es:

```
nombre_columna tipo_dato [NOT NULL | NULL] [DEFAULT valor_por_defecto]
    [AUTO_INCREMENT] [[PRIMARY] KEY] [COMMENT 'string']
    [definición_referencia]
```

Nota. MySQL 8.0 Reference Manual (2021)

Esta documentación se puede leer de la siguiente forma:

- **nombre_columna**: es el nombre que se le dará a la columna a definir.
- **tipo_dato**: es el tipo de datos de la columna (**INT, FLOAT, VAR CHAR**).
- **[NOT NULL | NULL]:** todo lo que encuentre encerrado entre corchetes (**[]**) significa que es opcional pude incluirse o no, dentro de las opciones.

  Lo que encuentre entre separado por el carácter pai (**|**), significa que debe escoger una opción de las que están paradas por el pai.
- De esta manera se quiere definir un de que se llame **fecha** de tipo **DATE** y que no sea nuleable la sintaxis sería:

```
fecha DATE NOT NULL
```

Si quiero que el valor por defecto de esta columna sea la fecha actual del sistema, puedo usar la sintaxis **DEFAULT CURRENT_DATE** así:

```
fecha DATE NOT NULL DEFAULT CURRENT_DATE
```

h) Claves primarias compuestas

La sintaxis para concretar claves primarias es:

```
definición_columnas
  | PRIMARY KEY (index_nombre_col,...)
```

Nota. MySQL 8.0 Reference Manual (2021)

Usando esta sintaxis, en el ejemplo anterior que se dio para crear claves primarias, quedaría de la siguiente forma:

```
mysql> CREATE TABLE ciudad6 (nombre CHAR(20) NOT NULL,
    -> poblacion INT NULL DEFAULT 5000,
    -> PRIMARY KEY (nombre));
Query OK, 0 rows affected (0.17 sec)
```

De esta manera, se tiene mayor cantidad de opciones, por ejemplo, entre los paréntesis se puedne detallar diversos nombres de columnas, para construir claves primarias acomodadas por diversas columnas:

```
mysql> CREATE TABLE persona (
    -> tipo_documento CHAR(2) NOT NULL,
    -> documento_identidad CHAR(10) NOT NULL,
    -> nombres CHAR(30),
```

```
   -> PRIMARY KEY (tipo_documento, documento_identidad));
Query OK, 0 rows affected (0.09 sec)
mysql>
```

Se puede observar como quedó definida la tabla, se debe prestar especial atención a la columna de salida llamada Key:

```
mysql> DESC persona;
+--------------------+----------+------+-----+---------+-------+
| Field              | Type     | Null | Key | Default | Extra |
+--------------------+----------+------+-----+---------+-------+
| tipo_documento     | char(2)  | NO   | PRI | NULL    |       |
| documento_identidad | char(10) | NO   | PRI | NULL    |       |
| nombres            | char(30) | YES  |     | NULL    |       |
+--------------------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

i) Índices

Se tienen tres tipos de índices. El primero pertenece a las claves primarias, que como se vió, también se pueden crear en la parte de definición de columnas.

¿Cuándo es oportuno crear índices?

a) Cuando se define una llave foránea.
b) Cuando se sabe que se realizarán consultas directas sobre esa columna.

El segundo tipo de índice que existe, permite definir sobre una columna el índice. Para concretar estos índices se usan de manera indistinta las opciones **KEY** o **INDEX**.

```
mysql> CREATE TABLE indices1 (
   -> id INT,
   -> nombre VARCHAR(20),
   -> INDEX (nombre));
Query OK, 0 rows affected (0.14 sec)
```

O su semejante:

```
mysql> CREATE TABLE indices2 (
   -> id INT,
   -> nombre VARCHAR(20),
   -> KEY (nombre));
Query OK, 0 rows affected (0.16 sec)
```

El tercer tipo de índices es para crear claves únicas así mismo sobre una columna o sobre varias. Para definir índices con claves únicas se debe de usar la opción **UNIQUE**.

La diferencia entre un índice único y un índice normal, es que en los índices únicos no se admite la inclusión de filas con datos repetidos en la columna o columnas de la tabla. Una excepción es el valor **NULL**, que sí se puede llegar a repetir varias veces.

```
mysql> CREATE TABLE indices3 (
   -> id INT,
   -> nombre CHAR(19),
   -> UNIQUE (nombre));
Query OK, 0 rows affected (0.09 sec)
```

Los índices son útiles para optimizar las consultas y las búsquedas de datos. Por medio de su uso es mucho más eficiente encontrar filas con ciertos valores de columnas, o mostrar los resultados en cierto orden. La alternativa es realizar búsquedas secuenciales (sin índices), que en tablas extensas requieren mucho tiempo y procesamiento de datos.