

# Manejo de Bases de Datos Relacionales con Python

## Indice

Introducción.....	1
Instalación de SQLite.....	2
SQLite Browser (funciones básicas):.....	2
Creación de tablas en una DB usando SQLbrowser.....	4
Buscar, modificar y borrar datos sobre una tabla con Sqlitebrowser.....	7
Bases de datos relacionales.....	10
Organización de la base de datos.....	11
Claves primarias.....	11
Claves foraneas.....	11
Integridad de los datos.....	13
Tipos de restricciones de integridad.....	13
Índices.....	14
Lenguaje SQL.....	15
Introducción.....	15
Crear tablas en lenguaje SQL.....	16
Modificar el esquema de una tabla en SQL.....	19
Borrado de tablas en SQL.....	21
Resumen Referencia SQL: Tablas.....	22
Operaciones sobre una tabla en lenguaje SQL.....	24
Consultas sobre una tabla.....	24
Insertar datos en una tabla.....	26
Actualizar datos en una tabla.....	26
Borrado de datos en tablas.....	27
Consultas sobre múltiples tablas.....	27
Consultas utilizando LEFT JOIN.....	27
Consultas utilizando INNER JOIN.....	28

## Introducción

En este curso, veremos cómo trabajar con bases de datos en Python.

Todo programa que sea medianamente útil, necesita cargar y guardar datos externos al mismo. Para esto, se almacenan los datos en bases de datos para su posterior consulta.

Si los datos son pocos y sin estructura, se pueden utilizar archivos. Pero cuando se necesita manejar grandes volúmenes de datos y además estos datos están relacionados unos con otros, utilizar archivos es inviable porque es poco eficiente y el código nos queda más complicado.

El tipo de base de datos más utilizado en la actualidad son las bases de datos relacionales. Sus bases fueron postuladas en 1970 por Edgar Frank Codd, en los laboratorios IBM en San José, California.

Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como un conjunto de datos llamados tuplas, con lo cual las **bases de datos pueden considerarse como un conjunto de relaciones o tablas que contienen registros o filas y cada registro contiene campos que son de un tipo de dato específico.**

La base de datos se organiza en dos secciones, el esquema y los datos:

- El esquema :es la definición de la estructura de la base de datos y principalmente almacena los siguientes datos:
  - el nombre de cada tabla,
  - el nombre de cada columna,
  - el tipo de dato de cada columna
  - y la tabla a la que pertenece cada columna.
- Los datos son el contenido de la base de datos en un momento dado. Es en sí el contenido de todos los registros.

Existen diversos sistemas utilizados para gestionar bases de datos relacionales. Por ejemplo, Oracle, SQL Server, PostgreSQL, SQLite, etcétera.

En este curso, utilizaremos el sistema SQLite, porque es muy liviano y fácil de instalar y configurar.

Virtualmente, todos estos sistemas de bases de datos relacionales utilizan el lenguaje SQL (*Structure, Query, Language*), para consultar y mantener la base de datos.

## Instalación de SQLite

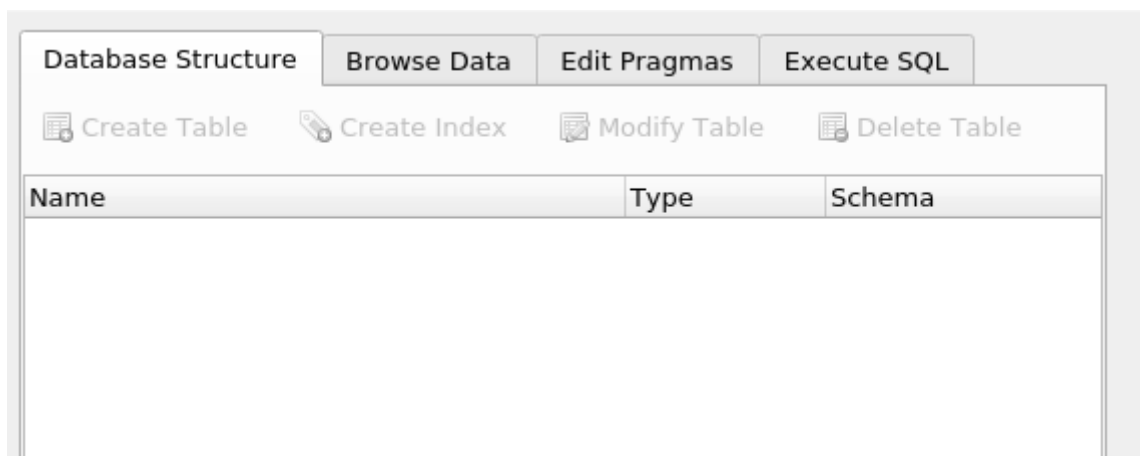
[https://d3c33hcgivew3.cloudfront.net/sdYG7o0pEemBQwq9JgxBCA\\_b39791f3ac07486ea39a6403aa965565\\_M1---L1---Instalacion-de-SQLite-y-SQLiteBrowser.pdf?Expires=1593907200&Signature=eJU~69xUkpRBEGM4-9v05erh7fwqB-PfaVOSPTjNIY6~6dlPaL4J0MtOvFtAIwDRROBhZII5ILCQ3x4I3RR9FZSISWqN9QAoLr0pyJLXAIXLBjrlkD0QBGv5pSf034GxwPszxUq-6688R2aIMe30ycTBMysGL2~D3jUjwRnwwDE\\_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A](https://d3c33hcgivew3.cloudfront.net/sdYG7o0pEemBQwq9JgxBCA_b39791f3ac07486ea39a6403aa965565_M1---L1---Instalacion-de-SQLite-y-SQLiteBrowser.pdf?Expires=1593907200&Signature=eJU~69xUkpRBEGM4-9v05erh7fwqB-PfaVOSPTjNIY6~6dlPaL4J0MtOvFtAIwDRROBhZII5ILCQ3x4I3RR9FZSISWqN9QAoLr0pyJLXAIXLBjrlkD0QBGv5pSf034GxwPszxUq-6688R2aIMe30ycTBMysGL2~D3jUjwRnwwDE_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)  
aparece un pdf que indica los pasos de instalación para windows, linux y mac.

## SQLite Browser (funciones básicas):

En la pantalla principal podemos ver cuatro pestañas:

- Database structure: nos sirve para modificar la estructura de la base de datos, es decir, crear, modificar o eliminar tablas, crear índices, "triggers", etcétera.
- Browse Data: nos permite buscar, insertar, modificar o eliminar datos en una tabla. En particular.
- Edit Pragmas: La tercera pestaña nos permite modificar algunos parámetros de la base de datos, como por ejemplo la cantidad máxima de páginas o el tamaño de una página, etcétera. Estas son opciones avanzadas de la base de datos.
- Execute SQL: nos permite ejecutar consultas SQL.

En la figura 1 se muestran las 4 pestañas:



*Figura 1: SQLite Browser: Aquí se muestran las 4 pestañas para trabajar con una base de datos SQL*

Por otra parte, en los menús de arriba (figura 2):

- "file" → "import": se nos permite importar una base de datos desde un archivo SQL, es decir, restaurar una copia de seguridad o importar una tabla desde un archivo CSV.
- "file" → "export", se nos permite exportar una base de datos a un archivo SQL, es decir, generar una copia de seguridad de la base de datos. También, se nos permite exportar una tabla a un archivo CSV o a un archivo JSON.



*Figura 2: Menú principal donde se pueden hacer montones de cosas que se van a ir viendo en el curso*

Lo primero que vamos a hacer es crear una nueva base de datos para ir trabajando durante el curso.

Esto se puede hacer desde el acceso directo en el menú o desde "file" → "new database". Vamos al directorio donde queremos crear la base de datos y le ponemos un nombre.

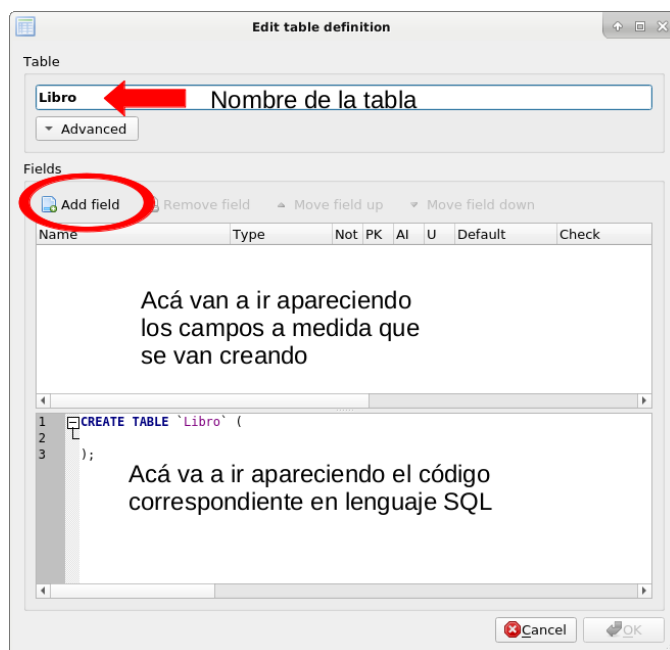
Cuando cerramos el programa, podremos volver a abrir la base de datos utilizando el acceso directo "open database" o desde "file" → "open database".

Hemos visto las opciones básicas que nos ofrece el navegador de base de datos SQLite Browser.

## Creación de tablas en una DB usando SQLbrowser

Para crear una nueva tabla hacemos clic en "create table". Se abre una nueva ventana para crear la tabla. En el primer campo, ponemos el nombre de la tabla.

En este ejemplo, crearemos la tabla libro: ver figura 3:



*Figura 3: Creación de una nueva tabla, en el campo señalado con la flecha roja, indica el nombre de la table, y en redondel rojo, la opción para agregar campos nuevos. A medida que se van creando o modificando los campos (columnas de la tabla SQL), se va a ir actualizando en el apartado de los campos y en el apartado del código en lenguaje SQL*

Entonces, escribimos libro.

Luego, tenemos que agregar los campos. En este caso, se agregarán cuatro campos, el ID, el ISBN, el título y la descripción.

Entonces, definimos los campos con sus tipos.

Para eso, vamos a presionar en Add field (Figura 3), le pondremos el nombre,

al primero le pondremos nombre ID, y le dejaremos el tipo Integer que es entero. Después, agregamos otro campo que será el ISBN del libro, que le pondremos como tipo, el tipo Text.

Luego, vamos a agregar otro field que va a ser el título, y le pondremos también tipo Text. Y por último, agregaremos un cuarto campo que será la descripción, que también será de tipo Text.

En realidad, SQLite no, no maneja tipos sino afinidades que terminan convirtiéndose al tipo más parecido. Digamos, si escribimos uno, dos, tres, se convertirá a entero. Pero no es que maneje internamente los tipos.

Acá el SQLite Browser nos permite elegir estos cinco tipos: Integer, Text, Blob, Real, Numeric. En el caso de Blob, es por ejemplo cuando queremos guardar una imagen o archivo, se puede usar este tipo de dato.

A la derecha, en las otras columnas, figuran algunas opciones más:

- **Not null**, se selecciona para que no se permita que ese campo esté vacío.

En este caso, vamos a definir como todas las columnas que no pueden estar vacías.

- **PK, o Primary Key**, que lo veremos más adelante.
- **AI**, que sería Autoincremental, que cada vez que se inserte una fila, se le asigna el número de fila más uno.

En este caso, va a ser el campo ID, que queremos que cada vez que se cree una, se inserte una nueva fila, se le ponga un campo autoincremental.

- **U**, que es de Único, que se le ha agregado un índice que indica que el campo no se puede repetir el valor en dos registros dentro de esa misma tabla. También lo veremos más adelante.
- **Default**: indica un valor por defecto a completar.

En el ejemplo, en el caso de descripción, podríamos agregar como valor por defecto el string vacío.

Abajo vemos el código SQL que genera la tabla. (Ver figura 4):

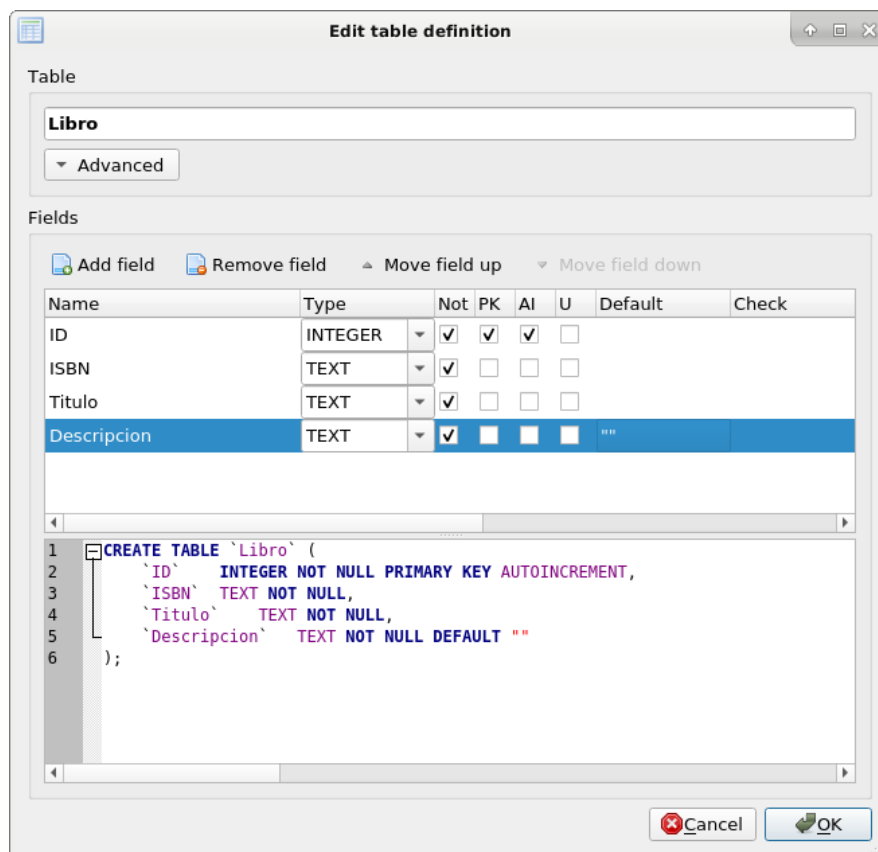


Figura 4: Tabla libro del ejemplo: Todos los campos no deben quedar vacíos, el campo ID es autoincremental (cada vez que se agrega una fila a la tabla, el número aumenta en 1) y la descripción lleva por defecto una cadena vacía. Abajo aparece la orden escrita en SQL

Presionamos el botón Okay, y ahora vemos la tabla creada con las columnas, con sus tipos, y su esquema, todo el esquema en SQL de la base, de la tabla, y de cada una de las columnas. (Figura 5)

Name	Type	Schema
Tables (2)		
Libro		CREATE TABLE `Libro` (
ID	INTEGER	`ID` INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
ISBN	TEXT	`ISBN` TEXT NOT NULL,
TITULO	TEXT	`TITULO` TEXT NOT NULL,
DESCRIPCION	TEXT	`DESCRIPCION` TEXT NOT NULL DEFAULT ''
sqlite_sequence		CREATE TABLE `sqlite_sequence` (

Figura 5: Base de datos donde se ve la tabla Libro con cada uno de sus campos. En la columna 2 se muestra el esquema; para cada uno de los campos de la tabla, en schema, se muestra el tipo de datos para cada uno de los campos.

Una vez que creamos la tabla, para que se guarden los cambios en disco, debemos presionar el botón Write Changes. (Figura 6)

Además, antes de escribir los cambios en disco, nos permite revertirlos presionando el botón “Revert Changes”. (Figura 6)

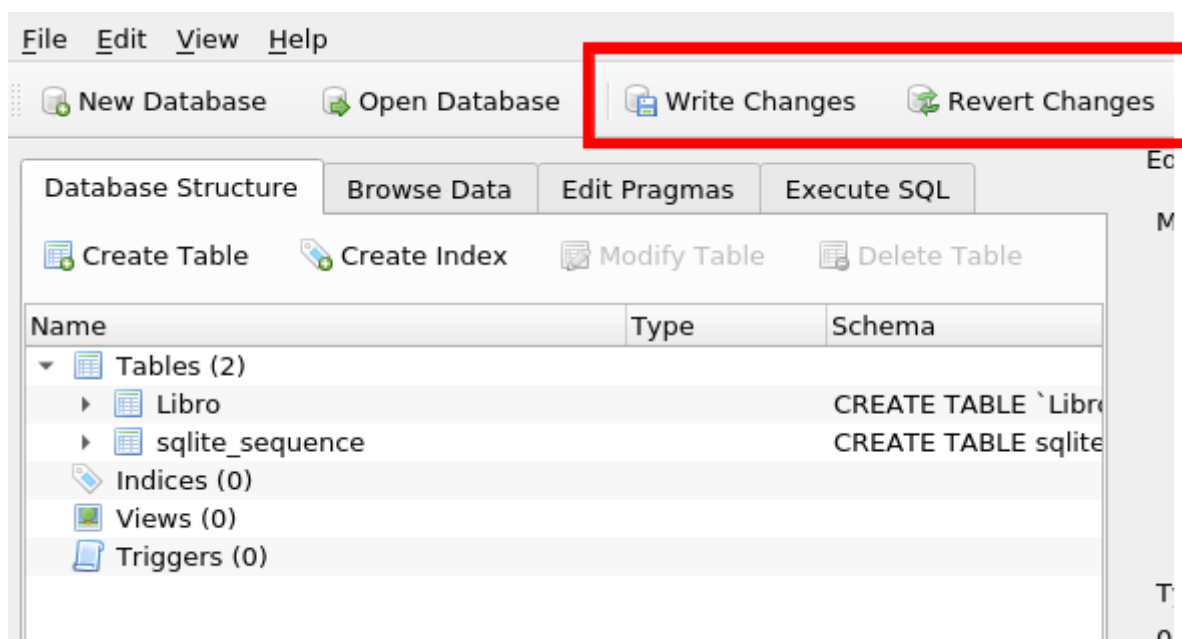


Figura 6: Guardar y revertir cambios: Ambos botones sirven para guardar y revertir cambios que se hayan hecho.

#### Notas:

- Es importante no olvidarse de guardar los cambios en disco, porque si se cierra el programa y no se guardan los cambios, los mismos se pierden.
- Cuando se quieren modificar el nombre de los campos, lo mejor es crear una tabla nueva pues el DB Browser se cierra y perdés los cambios

## Buscar, modificar y borrar datos sobre una tabla con Sqlitebrowser

Creamos el combo con las tablas, en este caso seleccionamos la tabla libro y vamos a agregar algunos registros a la tabla.

Para esto presionamos el botón new record, vemos que el campo id se selecciona con el 1 porque se autoincrementa, y después vamos a ir completando, el campo isbn, el isbn de un libro que será Primer Libro (Figura 7)

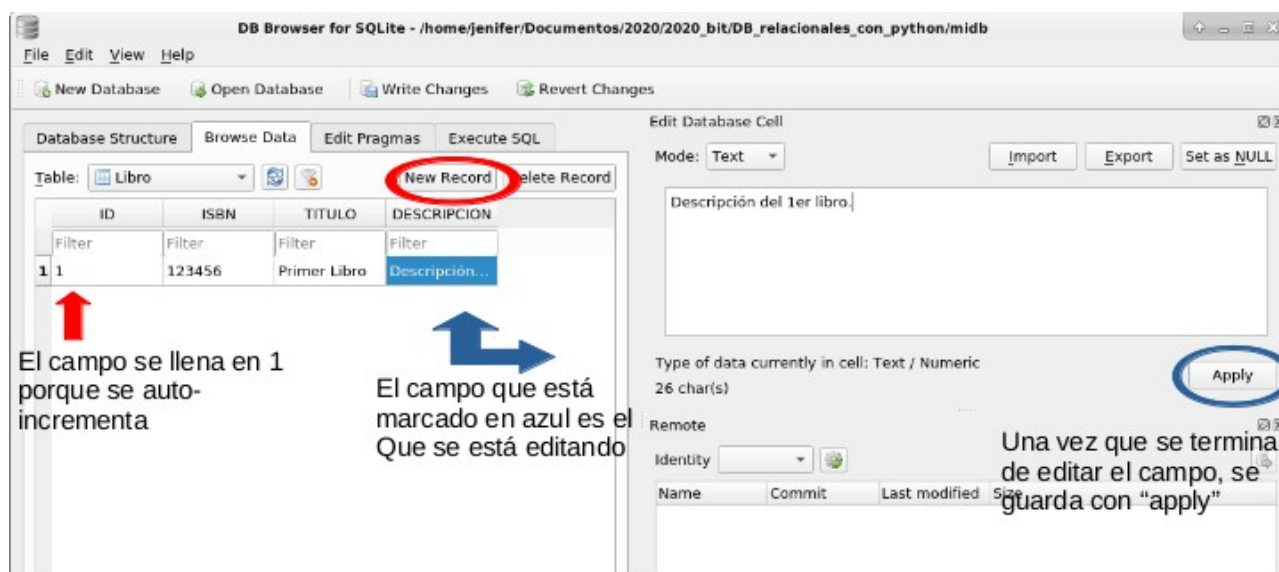


Figura 7: Creando un registro nuevo: Al presionar "New Record" se crea un registro nuevo. Para edita ese registro nuevo, se presiona sobre la celda a cambiar; a la derecha aparece un cuadro donde se puede editar la celda y se le da "apply". El campo ID al ser autoincremental, se completa automáticamente

Ahí vemos que se guarda y vamos a guardar la descripción también en el campo 3.

Luego, agregamos otro registro que va a ser el isbn de "Segundo libro" se pone el dato, y lo vamos a dejar sin descripción en este caso.

Haremos un tercer registro que será "Tercer Libro". Vamos a insertar el dato, por último la descripción y guardamos todos los datos. (Figura 8)

	ID	ISBN	TITULO	DESCRIPCION
	Filter	Filter	Filter	Filter
1	1	123456	Primer Libro	Descripción...
2	2	789012	Segundo Li...	
3	3	345678	Tercer Libro	Descripción...

Figura 8: Tabla Libro: aquí se muestran todos los libros guardados, ahí se ve claramente que ID es autoincremental y que para ID 2, el campo de descripción es una cadena vacía

Una vez que creamos los registros, podemos buscar por cualquiera de los campos, por ejemplo título, si buscamos "Se" vemos que filtra el libro "Segundo Libro" Lo mismo si voy filtrando el isbn, vamos filtrando el que corresponde. (Figura 9).





	ID	ISBN	TITULO	DESCRIPCION
	Filter	Filter	Se 	Filter
1	2	789012	Segundo Li...	

Figura 9: Filtrado: Si se busca en el campo "TITULO" la palabra "Se", como solo hay un registro que la contiene, muestra solo ese

Luego también podemos editar el campo, por ejemplo a la descripción ponerle no se campo test y aplicamos el cambio.

También podemos borrar un registro seleccionándolo y borramos el registro. Si se borran registros, NO se actualiza el id.(Ver Figura 10)

Table: Libro   New Record Delete Record

	ID	ISBN	TITULO	DESCRIPCION
	Filter	Filter	Filter	Filter
1	1	123456	Primer Libro	Descripción...
2	2	789012	Segundo Li...	test
3	3	345678	Tercer Libro	Descripción...

Se  
selecciona  
apretando  
aquí



	ID	ISBN	TITULO	DESCRIPCION
	Filter	Filter	Filter	Filter
1	1	123456	Primer Libro	Descripción...
2	3	345678	Tercer Libro	Descripción...

Aquí se nota el cambio pero no actualiza ID,  
si se agrega un registro nuevo, el id va a ser 4

Figura 10: Borrado de registros: para seleccionar un registro entetro (fila), se aprieta sobre el borde, en el lugar donde está el registro y se le da "delete record" (indicado con óvalo rojo)

También para que queden todos los datos en disco, hay que presionar el botón write changes.

## Bases de datos relacionales

El modelo relacional es el modelo de datos utilizado para modelar y gestionar bases de datos relacionales. Este modelo de datos está basado en la lógica de predicados y en la teoría de conjuntos.

Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. Con lo cual, las bases de datos relacionales se pueden considerar como un conjunto de relaciones (o tablas) que contienen registros (o filas) y cada registro está compuesto por campos que almacenan valores de un tipo de datos específico.

El modelo relacional asegura que cada registro de una tabla sea único, esto se hace creando una **clave primaria** (del inglés Primary Key). Con lo cual, no se permite tener dos registros con la misma clave primaria en una tabla.

Se llama **clave primaria** a un campo o a una combinación de campos que identifica de forma única a cada fila de una tabla. Una clave primaria comprende de esta manera una columna o conjunto de columnas. No puede haber dos filas en una tabla que tengan la misma clave primaria. Por ejemplo, la cédula de identidad de una persona o el ISBN de un libro son candidatas a ser una clave primaria.

A su vez cada tabla puede estar relacionada con otras tablas. Esta relación se realiza con las **claves foráneas** (del inglés Foreign Key). Una clave foránea es una restricción referencial entre dos tablas. La clave foránea identifica una columna o grupo de columnas en una tabla (**tabla referendo**) que se refiere a una columna o grupo de columnas en otra tabla (**tabla referenciada**). Las columnas en la tabla referendodeben ser la clave primaria u otra clave candidata en la tabla referenciada.

Los valores en una fila de las columnas referendo deben existir solo en una fila en la tabla referenciada. De esta manera, una fila en la tabla referendo no puede contener valores que no existen en la tabla referenciada. con lo cual, las referencias pueden ser creadas para vincular o relacionar información.

Además, múltiples filas en la tabla referendo pueden hacer referencia, vincularse o relacionarse a la misma fila en la tabla referenciada. En general, esto se ve reflejado en una relación uno (tabla referenciada) a muchos (tabla referendo).

3La tabla referendo y la tabla referenciada pueden ser la misma, esto significa que, la clave foránea remite o hace referencia a la misma tabla. A su vez, una tabla puede tener múltiples claves foráneas y cada una puede tener diferentes tablas referenciadas. Debido a que el sistema de gestión de base de datos hace cumplir las restricciones de referencia, se debe garantizar la integridad de los datos si las filas de la tabla referenciada se van a eliminar (o van a ser actualizadas). Si todavía existen filas dependientes en tablas referendo, esas referencias tienen que ser consideradas.

## Organización de la base de datos

La base de datos se organiza en dos secciones: el esquema y los datos

En cuanto al esquema, es la definición de la estructura de la base de datos y principalmente almacena los siguientes datos:

- El nombre de cada tabla
- El nombre de cada columna
- El tipo de dato de cada columna
- La tabla a la que pertenece cada columna

En cuanto a los datos, son el contenido de la base de datos en un momento dado. Es en sí, el contenido de todos los registros.

Los sistemas de software utilizados para mantener las bases de datos relacionales son conocidos como relational database management system (RDBMS) o sistema de gestión de bases de datos relacionales.

Si bien existen muchos sistemas de gestión de base de datos, todos utilizan (virtualmente) el lenguaje SQL para consultar y mantener la base de datos

## Claves primarias

La clave primaria es una restricción que identifica unívocamente cada registro en una tabla.

La misma debe contener valores únicos y no puede contener valores nulos; y una tabla puede tener una única clave primaria.

En la tabla la clave primaria consiste en una o múltiples columnas. **(esto no lo entendí)**

Si bien ciertos modelos pueden tener campos que sean posibles candidatos a ser la clave primaria de una tabla, como por ejemplo el ISBN del libro o la cédula de identidad, muchas veces es preferible crear un campo id que sea autoincremental y definir a este como la primary key de la tabla.

Y en el caso de ISBN que es único le creamos una restricción de unicidad. De esta manera después tendremos más facilidad para cambiar la composición de las tablas si es necesario.

La clave primaria también será muy importante para referenciar tablas, esto lo veremos más en detalle cuando veamos las claves foráneas.

En el ejemplo la tabla libro tiene un campo id que no es nulo, es primary key y es autoincremental. (ver figuras anteriores)

## Claves foráneas

Una clave foránea es una clave utilizada para relacionar o vincular dos tablas.

La misma es un campo o una colección de campos en una tabla que referencia a la clave primaria de otra tabla. La tabla que tiene definida la clave foránea es llamada "tabla hija", o "referendo", y la tabla que contiene la clave candidata es llamada "tabla padre" o "referenciada".

Pr ejemplo, dentro de la base de datos Primero, vamos a crear la tabla "autor" que tendrá los campos "ID", que es un entero, no nulo, que sería la "primary key" de la tabla "autor" y será autoincremental, y después los campos "NOMBRE" y "APELLIDO", que serán de tipo texto y serán no nulos. (Ver figura 11).

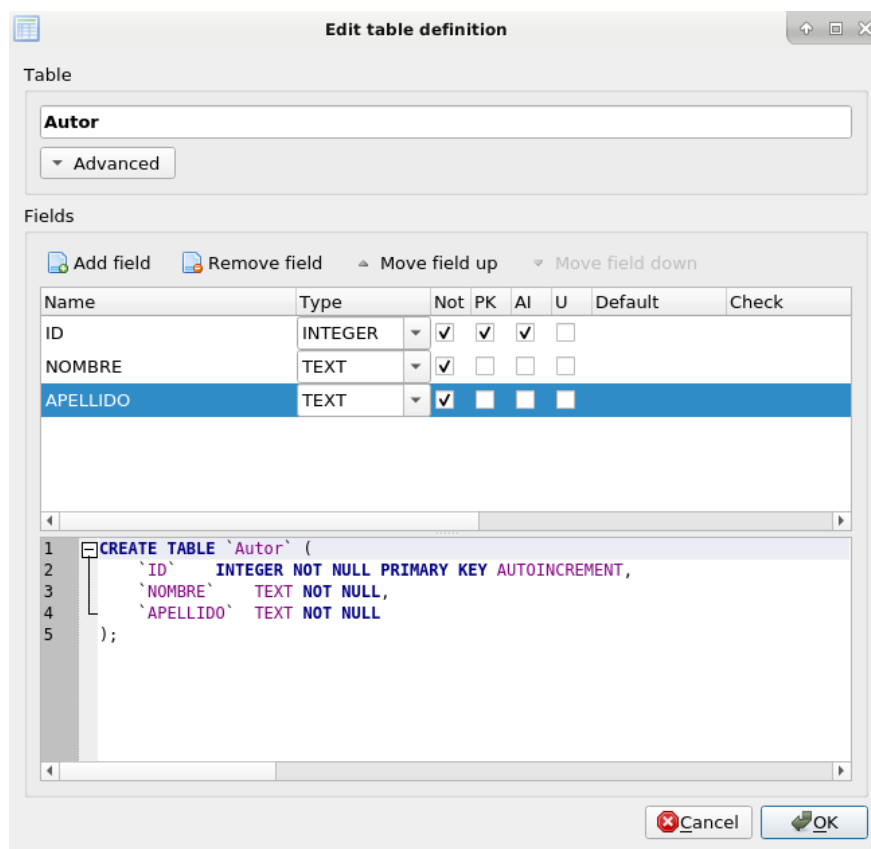


Figura 11: Ejemplo de tabla padre: se crea una tabla "Autor" donde el Primary Key es el ID, que es no nulo y autoincremental. Nombre y apellido son no nulos

Luego, iremos a la tabla "libro" y vamos a modificarla, agregándole un nuevo campo que será "AUTOR" que será de tipo "integer" porque será una referencia al "ID" de la tabla "autor" y luego, vamos a ir a la derecha, tenemos la columna vamos a ir a la derecha, tenemos la columna "Foreign Key" que la modificaremos para que apunte a la tabla "autor" y al campo "ID". Entonces, vemos que acá se autocompletó la columna "autor" que es entero y se crea una "foreign key" que referencia a la tabla "autor" en el campo "ID". (no lo muestro en figura porque me descompagina todo).

### Qué pasa cuando eliminamos un registro de la tabla referenciada/padre

En la especificación de SQL del año 2003, define cinco acciones referenciales diferentes que se ejecutarán en los casos de actualización o eliminación de registros de una tabla referenciada.

- "CASCADE". Cada vez que se eliminan o se actualizan las filas de una tabla padre, se eliminan o actualizan las respectivas filas de la tabla hija, esto se conoce como "eliminación" o "actualización en cascada".
- "RESTRICT" hace que el valor de una columna de la tabla padre no pueda ser actualizado o borrado cuando existe una fila en una tabla hija que hace referencia al valor de la columna de la tabla padre. A su vez, una fila no se puede eliminar, siempre que haya una referencia a la misma a partir de una tabla referendo.
- "NO ACTION" es similar a "RESTRICT": La principal diferencia entre el "NO ACTION" y "RESTRICT" es que "NO ACTION" realiza la comprobación de integridad referencial después de tratar de modificar la tabla, mientras que "RESTRICT" hace la comprobación antes de intentar ejecutar la sentencia "UPDATE" y "DELETE". Ambas acciones referenciales actúan de la misma manera, si la comprobación de integridad referencial falla, la sentencia "UPDATE" o "DELETE" dará un error.  
Ambas acciones referenciales (RESTRICT y NO ACTION) actúan de la misma forma si la comprobación de integridad referencial falla: la actualización o el borrado dará lugar a un error.
- "SET DEFAULT" hace que el valor de la referencia afectada se cambie con el valor predeterminado especificado.
- "SET NULL" hace que el valor de referencia afectada se cambie a "NULL". Para esto, las columnas de la tabla hija NO tienen que estar definidas como "NOT NULL".  
En general, la acción tomada por el sistema de gestión de base de datos para SET

NULL o SET DEFAULT es el mismo tanto para borrar como para actualizar. El valor de la referencia afectada se cambia a NULL para SET NULL, y con el valor predeterminado especificado por SET DEFAULT.

Con SET NULL, cada vez que se elimina o actualiza el registro en la tabla referenciada, establece a NULL las columnas de la clave foránea en la tabla referendo, para ello las columnas de la tabla referendo NO deben estar definidas como NOT NULL.

## Integridad de los datos

Se define como la correctitud y completitud de la información en la base de datos.

Cuando se modifica el contenido de la base de datos y sea modificando, agregando o borrando datos, la integridad puede perderse de diversas maneras por ejemplo poner datos inválidos (en nuestro caso ejemplo, poner un libro cuyo autor no esté en la DB), o modificarse datos existentes con un valor incorrecto (en nuestro ejemplo, si se reasigna a un libro una categoría que no existe)

Los cambios en la BD también pueden perderse por fallas en el sistema, como cortes de luz.

## Tipos de restricciones de integridad

### Integridad de dominio

Es la validez de las restricciones que debe cumplir una determinada columna de una tabla. Por ejemplo:

- **Datos requeridos:** Establece que una columna no tenga un valor nulo

- **Chequeo de validez:** cuando se crea una tabla, cada columna tiene la tabla tiene un tipo de datos, y el sistema de gestión de bases de datos asegura que solamente los datos del tipo especificado, sean insertados en la tabla.

### **Integridad de entidad**

Establece que la clave primaria de una tabla debe tener un valor único para cada fila de la tabla; si no, la base de datos perdería su integridad.

El sistema de base de datos comprueba automáticamente la unicidad del valor de la clave primaria cada vez que se inserta o se actualiza un registro, con lo cual, un intento de insertar un valor de clave primaria ya existente, dará error.

### **Integridad referencial**

Asegura la integridad entre las claves foráneas y primarias (relaciones tabla referenciada/tabla referendo).

Existen cuatro actualizaciones de la base de datos que pueden corromper la integridad referencial:

1. La inserción de una fila en la tabla referendo cuando no coincide la clave foránea con la clave primaria de la tabla referenciada.
2. La actualización de la clave foránea de una fila en la tabla referendo, cuando el valor de la misma no coincide con ninguna clave primaria en la tabla referenciada.
3. El borrado de una fila en la tabla referenciada, cuando tiene una o más filas en a tabla referendo que referencian a la fila borrada. En ese caso las filas de la tabla referendo quedarán huérfanas.
4. La actualización de la clave primaria de una fila en la tabla referenciada, cuando tiene una o más filas en la tabla referendo que referencian a la fila modificada. En ese caso las filas en la tabla referendo quedarán huérfanas.

Para asegurar la integridad referencial, SQL:2003 especifica 5 acciones referenciales diferentes que se ejecutarán en los siguientes casos: “CASCADE”, “RESTRICT”, “NO ACTION”, “SET DEFAULT” O “SET NULL” (Ver mas arriba en la sección “Qué pasa cuando eliminamos un registro de la tabla referenciada/padre” en el apartado “Claves foraneas”)

## **Índices**

Un índice es una estructura de datos que mejora la velocidad de las operaciones permitiendo un acceso rápido a los registros de una tabla en una base de datos.

Los índices pueden ser creados a partir de una o más columnas, proporcionando la base tanto para búsquedas rápidas al azar o el acceso ordenado a los registros de manera eficiente.

Además, los índices pueden ser definidos como únicos. Un índice único actúa como una restricción en la tabla previniendo filas idénticas en el índice.

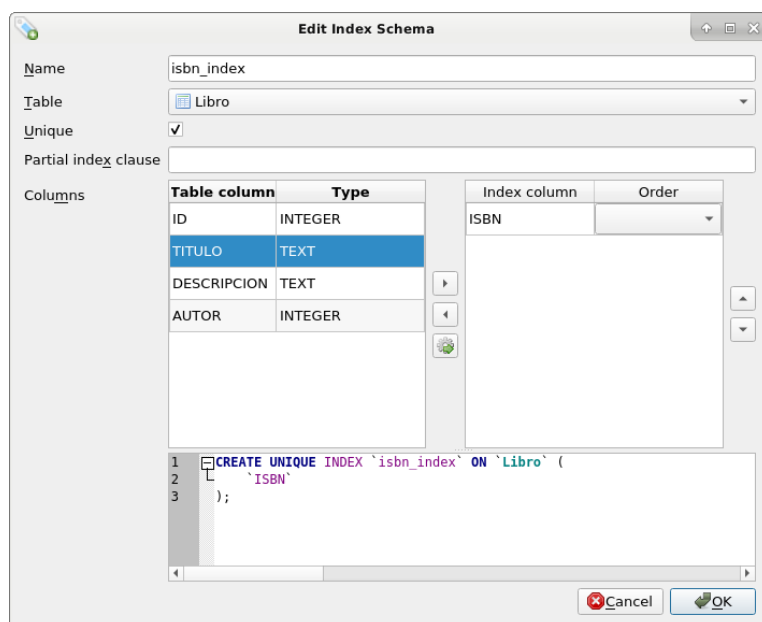
El índice ocupará un espacio en disco, generalmente, menor a la tabla que se indexa. Con lo cual, se consume más espacio en disco para ganar en velocidad en las consultas de búsqueda que es lo que se hace con más frecuencia en ciertas tablas de la base de datos.

Por otro lado, un índice, puede hacer más lenta la inserción, actualización y borrado de un registro porque además de tener que actualizar el dato en la tabla tiene que actualizar el índice. Con lo cual, conviene crear índices en tablas que son muy consultadas y poco modificadas.

Cabe aclarar, como detalle de implementación, que los índices son construidos generalmente sobre árboles B, B+ o B\* (ÉSTO TIENE QUE VER CON ALMACENAMIENTOS DE DATOS EN MEMORIA EN FORMA DE ÁRBOL [https://en.wikipedia.org/wiki/B%2B\\_tree#Search](https://en.wikipedia.org/wiki/B%2B_tree#Search)).

### Creación de un índice en el Dbbrowser

QUEDA PARA EDITAR DESPUÉS



Comando SQL en general:

```
CREATE UNIQUE INDEX `<nombre_del_indice>` ON `<nombre_de_la_tabla>` (
    `<nombre_de_la_columna>`, ← va la coma cuando hay mas de una columna
    <nombre_de_otra_columna>`
);
```

## Lenguaje SQL

### Introducción

El lenguaje SQL, en inglés Structured Query Language o Lenguaje de Consulta Estructurada, es un lenguaje diseñado para administrar y recuperar información de sistemas de gestión de bases de datos relacionales. Es el lenguaje de base de datos más utilizado.

Una de sus principales características es el manejo del álgebra y cálculo relacional para realizar consultas, con el fin de obtener información de la base de datos y realizar cambios en ella.

SQL consiste en un lenguaje de definición, manipulación y control de datos. Su alcance incluye:

- la inserción, consulta, actualización y borrado de datos
- la creación y modificación de esquemas
- el control de acceso a los datos.

Algunas características de SQL son:

- el lenguaje de definición de datos incluye comandos para la definición de esquemas de relación, borrado de relaciones y modificaciones de los esquemas de relación
- el lenguaje de manipulación de datos incluye lenguajes de consulta basados tanto en álgebra relacional como en el cálculo relacional de tuplas
- permite definir la integridad de los datos, es decir, incluye comandos para especificar restricciones de integridad que deben cumplir los datos almacenados en la base de datos;
- permite definir vistas,
- permite trabajar con transacciones, incluye comandos para especificar el comienzo y el final de una transacción, incluye comandos para especificar los derechos de acceso a las relaciones y las vistas.

Algunos de los tipos de datos básicos de SQL son:

- VARCHAR que es una cadena de texto compuesta de letras, números y caracteres especiales; "
- integer", que son valores enteros con o sin signo;
- date que es una fecha de calendario que contiene el año, el mes y el día;
- "time" que es la hora del día en horas, minutos y segundos.

## Crear tablas en lenguaje SQL

La sentencia CREATE TABLE sirve para crear tablas nuevas, Su sintaxis es

```
CREATE TABLE `<Nombre_de_la_tabla>` (  
  `<Columna>` `<tipo_de_dato>` NOT NULL* PRIMARY KEY** AUTOINCREMENT***,  
  `<Columna>` `<tipo_de_dato>` NOT NULL DEFAULT <valor_por_defecto>,  
  `<nombre_del_campo_a_ser_clave_foranea>` `<tipo_de_dato_usualmente_INTEGER>`,  
  FOREIGN KEY(`<nombre_del_campo_a_ser_clave_foranea>`) REFERENCES  
  `<Nombre_de_la_tabla_padre>`(`<campo_referenciado****>`)  
);
```



\*Si corresponde que sea NOT NULL

\*\*Si ese campo es clave primaria

\*\*\* Si corresponde que sea autoincremental

\*\*\*\* Es la clave primaria de la tabla padre.

O sea es "CREATE TABLE", el nombre de la tabla y, después, se definen las columnas con su tipo de dato.

Además, al definir las columnas en la creación de la tabla se pueden especificar algunas restricciones de esas columnas.

Las restricciones posibles son:

- "NOT NULL" que asegura que la columna no tenga el valor "NULL"
- "UNIQUE" asegura que todos los valores en una columna sean distintos
- "PRIMARY KEY", una combinación de "NOT NULL" y "UNIQUE", identifica unívocamente cada registro en una tabla;
- "FOREIGN KEY", se identifica de forma única un registro en otra tabla;
- "CHECK" asegura que todos los valores en una columna satisfacen una condición específica, por ejemplo, que sean valores mayores a "cero";
- "DEFAULT", pondrá un valor por defecto para una columna cuando no se especifica un valor;
- "INDEX" son utilizados para crear y devolver datos en forma eficiente, no es soportado por todas las versiones de "SQL".
- "AUTOINCREMENT". La misma genera automáticamente un número único incremental cuando se inserta un nuevo registro a la tabla.

Muchas veces se utiliza en los campos que son la clave primaria de la tabla.

Veamos un ejemplo.

Vamos a crear una tabla llamada "categoría\_libro", que tendrá un código que será autoincremental, no nulo, de tipo entero y será la clave primaria de la tabla, y un campo "NOMBRE", que va a ser de tipo texto y no nulo (ver figura 12).

Entonces, nos posicionamos en la consulta y vamos a ejecutarla. Ahí vemos que se ejecutó y si vemos la estructura de base de datos, vemos que se creó la tabla "categoría\_libro", que tiene dos campos, "CÓDIGO", que es no nulo, clave primaria autoincremental de tipo entero, y tiene el campo "NOMBRE", que es de tipo texto y es no nulo.

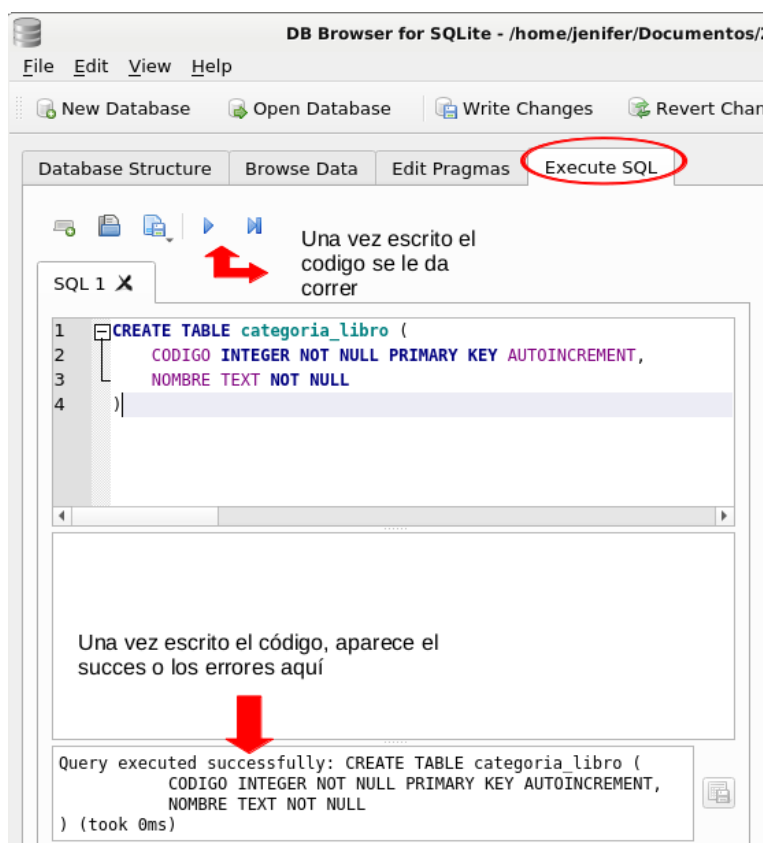


Figura 12: Ejecutando código SQL: en la pestaña de "Execute SQL" se escribe el código en el recuadro donde aparece y luego se le da a la flecha de play (marcada en rojo). En el tercer recuadro aparece, una vez que se corrió, el mensaje de success o los errores de sintaxis.

Cuando hay más de una consulta, se posiciona el cursor sobre la consulta, y se le da al botón de play parcial (el que está al lado de play señalado)

Además, con la sentencia "CREATE" se puede crear una copia de una tabla existente. La nueva tabla tendrá la misma definición de columnas, se podrán seleccionar todas las columnas o algunas específicas.

Si se crea una tabla nueva a partir de una tabla existente, la nueva tabla sesionará con los valores existentes de la tabla vieja.

La sintaxis es: "CREATE TABLE", el nombre nuevo de la tabla, "AS", y se hace un "SELECT" de las columnas que uno quiere incorporar en la tabla nueva desde la tabla vieja, y se le pueden poner condiciones para filtrar ciertos valores.

```
CREATE TABLE AS SELECT <columna_1>...<columna_n> FROM  
<nombre_de_la_tabla_a_copiar_los_valores> WHERE*
```

\*El where es opcional, se usa para seleccionar silo aquellos registros que cumplan on una condición.

En este caso, vamos a crear una tabla "libro\_copy" y la crearemos desde la tabla "libro" pero con las columnas "ISBN" y "TÍTULO". (Figura 13).

Entonces, ejecutamos esta consulta. Ahora, vamos a la estructura y vemos que se creó la tabla "libro\_copy" con los campos "ISBN" y "TÍTULO". Ahora, si inspeccionamos los datos que tiene la tabla "libro\_copy", vemos que tiene dos filas, que son el libro de "Primer libro", "Segundo libro" y el "Tercer libro", que son los mismos datos que tiene la tabla "libro", nada más que sólo con las columnas "ISBN" y "TÍTULO".

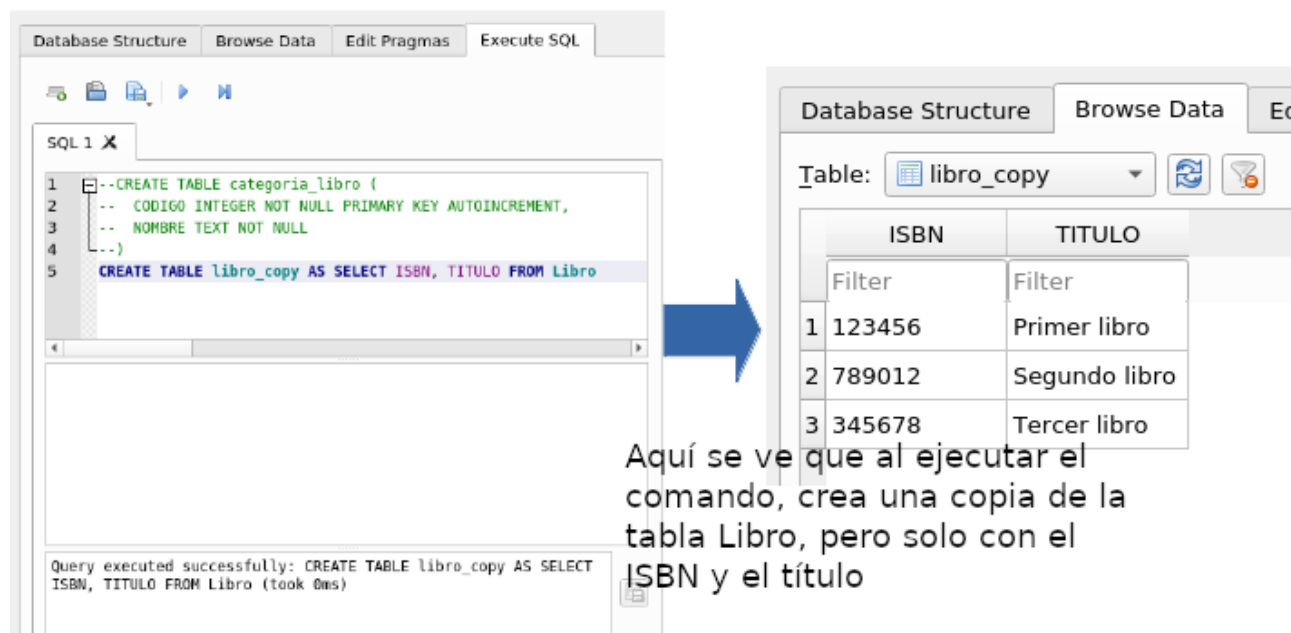


Figura 13: Creado de copia parcial de una tabla: aquí se vé como se usa *SELECT* para copiar sólo el ISBN y el Título de los libros archivados en la tabla Libro

## Modificar el esquema de una tabla en SQL

Para modificar el esquema de una tabla, se realiza con el comando "**ALTER TABLE**".

La sentencia "alter table" se utiliza para agregar, borrar o modificar columnas en una tabla existente. Además, esta sentencia se utiliza para agregar o quitar restricciones, sobre las columnas de una tabla existente.

- **Para agregar una columna** se utiliza la siguiente sintaxis:  
"ALTER TABLE", el nombre de la tabla, y se agrega la columna ("ADD") con el nombre "column\_name" y el tipo de dato.  
Entonces, por ejemplo, podemos agregar en la columna "precio" de tipo entero, a la tabla "libro", si ejecuto vemos que ahora la tabla "libro", tiene la columna "precio" de tipo entero.  
ALTER TABLE Libro ADD precio INTEGER; (ver figura 14)

```
ALTER TABLE Libro ADD precio INTEGER;
```



Si se ejecuta éste código, se ve que se agrega una nueva columna con el precio

precio

Table:	<div>Libro</div>	<div><div></div><div></div></div>	<div>New Record</div>	<div>Delete Record</div>	
	ISBN	TITULO	DESCRIPCION	AUTOR	precio
	Filter	Filter	Filter	Filter	Filter
1	123456	Primer libro	Descrip.pri...	1	NULL
2	789012	Segundo libro		1	NULL
3	345678	Tercer libro	Decrip. Terc...	2	NULL

Figura 14: Agregar columnas: Se agrega la columna precio que es un integer

También puedo agregar una columna y, además, con una "foreignKey" que referencia a otra tabla:

Entonces, por ejemplo, agrego en la tabla "libro", la columna "categoría\_id", que es entero y que referencia a la "id" de la tabla categoría "libro".

Si ejecuto la consulta, vemos que ahora la tabla "libro", tiene un campo "categoría\_id" y a la vez si veo la tabla, vemos que tiene una "foreign key" a la tabla categoría "libro", apuntando al campo "id". (Ver Figura 15 )

```
ALTER TABLE Libro ADD categoria_id  
INTEGER REFERENCES Categoria_libro(ID)
```

Figura 15: Ejemplo de agregado de una columna con una clave foranea: En ésta figura, toma la tabla Libro, le agrega la columna categoria\_id que es de tipo entero y va contener claves foraneas (REFERENCES), tomadas de la tabla Categoría\_libro, siendo el campo de dicha tabla padre, el campo ID

- **Para borrar una columna**, utilizar la siguiente sintaxis: "ALTER TABLE", el nombre de la tabla y "DROP COLUMN", el nombre de la columna. (La sintaxis se ve en la Figura 16) No todas las versiones "SQL" lo permiten, en este caso, no lo permite

```
ALTER TABLE nombre_de_la_tabla  
DROP nombre_de_la_columna;
```

Figura 16: Sintaxis para borrar columna: Va "ALTER TABLE" luego el nombre de la tabla a modificar, luego "DROP" y después el nombre de la columna a borrar

- **Para modificar una columna, el tipo de dato o una restricción** se utiliza "alter table" el nombre de tabla, "ALTER COLUMN", el "column name" y el tipo de dato. En este caso, también hay versiones "SQL" que no lo permiten (Ver sintaxis en Figura 17)

```
ALTER TABLE nombre_de_la_tabla ALTER COLUMN  
nombre_de_la_columna INTEGER;
```

Figura 17: Sintaxis para modificar una columna:columna: Va "ALTER TABLE" luego el nombre de la tabla a modificar, luego el nombre de la columna y luego el tipo de dato, que en el caso de la imagen es INTEGER pero puede ser otro

- **Para agregar una restricción**, se puede usar "ALTER TABLE", la tabla y "ADD CONSTRAINT", el nombre da la restricción. Se puede decir, por ejemplo, "CHECK", "precio>=0". (Ver figura 18). En este caso, no está permitido establecer en "SQL", pero gran parte de las versiones de "SQL" permiten este tipo de consultas.

```
ALTER TABLE Libro ADD CONSTRAINT CHECK (PRECIO >0)  
  
ALTER TABLE Persons  
ADD CONSTRAINT PK_Person PRIMARY KEY (ID,LastName);
```

Figura 18: Añadiendo constrains: Siempre va "ALTER TABLE", luego el nombre de la tabla, después "ADD CONSTRAINT", luego un nombre significativo de la restricción y luego la restricción correspondiente.

En el primer caso, la tabla se llama Libro y la restricción es que la columna precio debe ser mayor que 0

En el segundo caso, el nombre de la restricción es PK\_Person y el constrain es que los campos ID y LastName sean claves primarias de la tabla Person (es así o entendí mal?)

## Borrado de tablas en SQL

La sentencia "**DROP TABLE**" se utiliza para borrar una tabla existente en la base de datos. La sintaxis es: "DROP TABLE" y el nombre de la tabla

Algunas versiones de SQL proveen la sentencia "TRUNCATE TABLE", que se utiliza para borrar los datos de una tabla, pero no la tabla misma.

La sintaxis es: "TRUNCATE TABLE" y el nombre de la tabla.

## Resumen Referencia SQL: Tablas

En esta sección se presenta una referencia de las sentencias de SQL para crear, modificar y eliminar una tabla de la base de datos.

### Creación de tablas

La sentencia CREATE TABLE se utiliza para crear una tabla nueva en la base de datos.

La sintaxis básica Se muestra en la Figura 19

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    ....  
);
```

*Figura 19: Sintaxis para crear una tabla: "CREATE TABLE", el nombre de la tabla, parentesis curvos, el nombre de cada columna y su tipo de dato, separada cada columna con coma, y al cerrar paréntesis va después un punto y coma*

Los tipos de datos que maneja SQLite son INTEGER, TEXT, NUMERIC, REAL, BLOB

Además, al definir las columnas en la creación de una tabla se pueden especificar algunas restricciones de esas columnas:

- NOT NULL: Asegura que la columna no tenga el valor NULL. Con lo cual, la columna tendrá que tener siempre definido algún valor.
- UNIQUE: Asegura que todos los valores en una columna sean distintos.
- PRIMARY KEY: Una combinación de NOT NULL y UNIQUE. Identifica unívocamente cada registro en una tabla.
- FOREIGN KEY: Se identifica de forma única un registro en otra tabla.
- CHECK: Asegura que todos los valores en una columna satisfacen una condición específica. Por ejemplo, que sean valores mayores a cero.
- DEFAULT: Pone un valor por defecto para una columna cuando no se especifica un valor.
- INDEX: Utilizados para crear y devolver datos de forma eficiente. No soportado, en la creación de la tabla, por todas las versiones de SQL.
- AUTOINCREMENT: Genera automáticamente un número único (incremental) cuando se inserta un nuevo registro a la tabla. Muchas veces se utiliza en los campos que son la clave primaria de la tabla.

### Creación de una tabla a partir del resultado de una consulta

Además, con la sentencia CREATE TABLE se puede crear una copia de una tabla existente, como resultado de una consulta. La nueva tabla tendrá la misma definición de columnas. Se podrán seleccionar todas las columnas o algunas específicas.

Si se crea una tabla nueva a partir de una tabla existente, la nueva tabla se llenará con los valores existentes de la tabla vieja

La sintaxis básica se puede ver en la Figura 20:

```
CREATE TABLE new_table_name AS  
SELECT column1, column2,...  
FROM existing_table_name  
WHERE ....;
```

*Figura 20: Creación de tablas a partir de una consulta sobre una tabla vieja: CREATE TABLE, nombre de la tabla nueva, AS SELECT, las columnas a seleccionar separadas con coma, FROM luego nombre de la tabla de origen; y se usa WHERE para poner una restricción a la consulta*

### Modificación de tablas:

La sentencia ALTER TABLE se utiliza para agregar, borrar o modificar columnas en una tabla existente. Además, esta sentencia se utiliza para agregar o quitar restricciones sobre una tabla existente.

La sintaxis se puede ver en la Figura 21

```
-- Se agrega columna column_name  
ALTER TABLE table_name  
ADD column_name datatype;  
  
-- Se borra columna column_name  
ALTER TABLE table_name  
DROP COLUMN column_name;  
  
-- Actualizo columna column_name  
ALTER TABLE table_name  
ALTER COLUMN column_name datatype;
```

*Figura 21: Sintaxis para alterar columnas de una tabla. ADD agrega una columna nueva, DROP elimina una columna y ALTER COLUMN altera el tipo de datos de la columna deseada*

En la figura 22 se ven ejemplos:

```
ALTER TABLE libro
ADD categoria_id INTEGER REFERENCES categoria_libro(ID);

ALTER TABLE libro
DROP COLUMN categoria_id;

ALTER TABLE libro
ALTER COLUMN precio REAL;

-- Agregar una restricción
ALTER TABLE libro
ADD CONSTRAINT chk_price CHECK (precio>=0);
```

Figura 22: ejemplos de modificación de tablas.

La primera consulta agrega la columna *categoria\_id* de tipo *INTEGER* que referencia a la tabla *categoria\_libro* a través de la columna *ID*. Con lo cual, se está creando una columna a la vez que se define una clave foránea.

La segunda consulta borra la columna *categoria\_id*.

La tercera consulta actualiza el tipo de datos de la columna *precio* al tipo *REAL*.

La cuarta consulta agrega una restricción que verifica que la columna *precio* contenga valores mayores o iguales a cero

### Borrado de tablas

- La sentencia *DROP TABLE* se utiliza para borrar una tabla existente en la base de datos.

La sintaxis básica es: *DROP TABLE table\_name*;

Donde *table\_name* es el nombre de una tabla existente en la base de datos.

- La sentencia *TRUNCATE TABLE* se utiliza para borrar los datos de una tabla sin borrar la misma.

La sintaxis es: *TRUNCATE TABLE table\_name*;

Cabe aclarar que no todos los motores de base de datos soportan esta sentencia.

## Operaciones sobre una tabla en lenguaje SQL

### Consultas sobre una tabla

- *SELECT*: La sentencia select, se utiliza para seleccionar datos desde una base de datos. Los datos devueltos son guardados en un tabla de resultado llamada conjunto resultado. La sintaxis básica de un select es: *SELECT*, las columnas que uno quiera, *FROM*, el nombre de la tabla.



También se puede usar el asterisco para indicar que nos traiga todas las columnas de una tabla. Entonces, `SELECT * FROM el_nombre_de_la_tabla`.

- `SELECT DISTINCT`, se utiliza para devolver solo los valores diferentes.

Una columna de una tabla generalmente, contiene muchos valores duplicados y a veces solo queremos una lista de los valores distintos.

La sintaxis es, `SELECT DISTINCT, <las columnas que queremos seleccionar> FROM <el nombre de la tabla>`.

## Clausula WHERE

Por otra parte, se puede usar la cláusula where, que se utiliza para filtrar registros, es decir se utiliza para extraer solo los registros que cumplen la condición especificada.

La sintaxis es:

`SELECT, <las columnas que queremos seleccionar> FROM <nombre de la tabla> WHERE <condición que debe cumplir>`.

Ejemplo: `SELECT, * FROM Libro WHERE precio >=100`

Esto significa que va a seleccionar los libros donde el precio del libro sea mayor o igual a 100.

La cláusula where puede ser combinada con los operadores lógicos and, or y not.

Los operadores and y or, son utilizados para filtrar registros basados en más de una condición.

- **AND** muestra un registro si todas las condiciones separadas por el operador and, son verdaderas.

La sintaxis del and es, seleccionar las columnas desde la tabla cuando se cumpla la condición 1 y la condición 2 y la condición 3 (ver figura 23):

```
SELECT <columnas> FROM <tabla>
WHERE <condicion1> AND <condicion2> AND <condicion3>
```

*Figura 23: Sintaxis de AND: Los signos < y > no van, son solo para especificar que ahí va un parametro, por ejemplo <tabla> es nombre de la tabla*

Por ejemplo, seleccionar todas las columnas desde la tabla libro, cuando el precio está entre 90 y 200; en ese caso la sintaxis es:

`SELECT * FROM Libro WHERE precio > 90 AND precio < 200`

- **OR** muestra un registro si alguna de las condiciones separadas por el operador or, es verdadera. La sintaxis es similar a AND. (Figura 24)

```
SELECT <columnas> FROM <tabla>
WHERE <condicion1> OR <condicion2> OR <condicion3>
```

*Figura 24: Sintaxis de OR, se puede ver que es similar a la de AND*

Por ejemplo seleccionamos todas las columnas desde la tabla libro, cuando el precio es menor igual a 100 o el precio mayor a 200, en ese caso, la sintaxis es:

`SELECT * FROM Libro WHERE precio <=100 OR precio > 200`

- **NOT** muestra un registro si la condición no es verdadera. La sintaxis del NOT es:  
`SELECT <columnas> FROM <tabla> WHERE NOT <condicion>`

Por ejemplo, seleccionamos todas las columnas desde la tabla libro cuando la categoría no es la número 1. En ese caso la sintaxis es:

```
SELECT * FROM Libro WHERE NOT categoria_id = 1;
```

Para combinar operaciones se usan paréntesis curvos

Ej:

```
SELECT * FROM Customers WHERE Country='Germany' AND (City='Berlin' OR City='München');
```

## **ORDER BY**

La palabra clave order by, se utiliza para ordenar el conjunto resultado de forma ascendente o descendente. Por defecto, ordena los registros de forma ascendente.

Para ordenar los registros de forma descendente, usa la palabra clave DESC.

Entonces la sintaxis del order by es, seleccionar las columnas desde el nombre de la tabla y order by, las columnas por lo que quiere ordenar.

Se ordena primero por la columna 1, después por la columna 2 y así siguiendo y después se puede agregar ASC o DESC, según si es ascendente o descendente.

Por otra parte, si se quiere ordenar por distintos criterios, por ejemplo, si se tiene la tabla Customers (Clientes) y las filas “Country” (País) y “CustomerName” (Nombre del cliente) y se quiere ordenar por “Country” de forma ascendente y “CustomerName” de forma descendente, la sintaxis es:

```
SELECT * FROM Customers ORDER BY Country ASC, CustomerName DESC;
```

## **Limit**

También se puede limitar la cantidad de resultados que queremos devolver con la sentencia limit, seguido de la cantidad de registros a devolver como máximo.

Entonces por ejemplo, seleccionar todas las columnas desde la tabla libro y limit 1, significa que me traiga un solo registro.

## **Insertar datos en una tabla**

Se usa la palabra clave INSERT INTO, y se puede especificar de 2 maneras:

1. `INSERT INTO table_name (columna1, columna2, columna3, ...) VALUES (valor1, valor2, valor3, ...);` (acá va pareado columna1 con valor1 y así), acá no es necesario poner todos los valores, por ejemplo, si se tiene una columna que es autoincremental, no es necesario ponerla.
2. `INSERT INTO table_name VALUES (value1, value2, value3, ...);` Si se usa ésta manera, los valores se deben insertar en el mismo orden que aparecen en la tabla y se deben poner todos los campos. Cabe destacar que acá SI es necesario poner todos los valores, por ejemplo, si se tiene una columna que es autoincremental, hay que asignarla igual.

## **Actualizar datos en una tabla**

Para ésto se usa la sentencia UPDATE conjuntamente con la palabra SET:

```
UPDATE table_name SET column1 = value1, column2 = value2, ...WHERE condition;
```

donde condition se usa para elegir qué registros se quieren modificar, si no, modifica todos los registros con los valores especificados.

Por ejemplo, modificamos el nombre de la categoría de libros con código 4, asignándole como nuevo nombre “Acción”:

```
UPDATE categoria_libro SET NOMBRE = 'Acción' WHERE CODIGO = 4
```

Donde la palabra “NOMBRE” es el como se llama la columna a modificar y “WHERE CODIGO = 4” es que le ponga “Acción” en todos los registros donde el dato de la columna CODIGO sea igual a 4.

## Borrado de datos en tablas

Se usa la sentencia DELETE

```
DELETE FROM table_name WHERE condition;
```

Al igual que UPDATE, WHERE se usa para especificar los registros que se quieren borrar, si no se usa, borra todos los registros.

Por eso también notar que es posible borrar todos los registros en una tabla, sin borrar la tabla. Esto significa que la estructura de la tabla, en las columnas y los índices quedan intactos. Para eso se hace "DELETE FROM" y el nombre de la tabla, o sea, sería equivalente al “TRUNCATE TABLE”

## Consultas sobre múltiples tablas

La sentencia JOIN permite consultar datos de dos o más tablas. Dichas tablas están relacionadas entre ellas de alguna forma a través de alguna de sus columnas. El propósito de JOIN es unir información de diferentes tablas para no tener que repetir dos datos iguales en diferentes tablas.

Hay distintos tipos de join pero los más usados son LEFT JOIN e INNER JOIN

### Consultas utilizando LEFT JOIN

La palabra clave LEFT JOIN devuelve todos los registros desde la tabla izquierda (tabla A), y los registros de la tabla derecha (tabla B) que coinciden en el campo por el que se hace la unión. El resultado de los campos de la tabla derecha quedan en NULL cuando no coincide el campo por el que se hace la unión.

Su sintaxis se puede ver en la figura 25:

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

*Figura 25: Sintaxis de LEFT JOIN*

Por ejemplo, si queremos seleccionar los libros y tener el nombre de la categoría del libro entre las columnas seleccionadas, entonces podemos realizar la consulta mostrada en la figura 26:

```
SELECT * FROM libro L  
LEFT JOIN categoria_libro C  
ON L.categoria_id = C.CODIGO;
```

Figura 26: Ejemplo de LEFT JOIN. En ésta figura se usa L y C para hacer referencia a las tablas Libro y categoria\_libro respectivamente

## Consultas utilizando INNER JOIN

La palabra clave INNER JOIN selecciona registros que coinciden los valores en ambas tablas.

La Sintaxis se muestra en la figura 27.

```
SELECT column_name(s)  
FROM table1  
INNER JOIN table2  
ON table1.column_name = table2.column_name;
```

Figura 27: Sintaxis de INNER JOIN

La diferencia con el left join es que el inner join se queda con los registros que coinciden en ambas tablas.

En cambio el left join se queda con todos los registros de la tabla izquierda, coincidan o no con los valores de la tabla derecha, y si no coinciden, en lo que correspondería en la tabla derecha, pone NULL.