



Bases de Datos. Nivel I

Marzo, 2019



Objetivos del nivel

- Conocer los conceptos fundamentales de bases de datos.
- Conocer los tipos de relaciones entre las tablas.
- Aprender a leer un MER (Modelo Entidad Relación).
- Crear una base de datos física.
- Aprender a usar bases de datos SQLite.

Prerrequisitos del nivel

Este nivel del curso no tiene pre requisitos

Acerca de este manual

Este manual pertenece al Centro de Asesoramiento y Desarrollo Informático C.A. (CADIF1). Para obtener más información sobre este u otros cursos visite nuestra sitio Web www.cadif1.com, escribanos a la dirección de correo cadi@cadif1.com o visítenos en nuestra sede ubicada en la Av. Pedro León Torres con calle 59, Centro Comercial Sotavento, piso 2 oficina 27, Barquisimeto estado Lara, Venezuela. Tlf. 0251-7179247, 0251-4410268.

Las marcas mencionadas en este manual son propiedad de sus respectivos dueños.
Copyright 2019. Todos los derechos reservados.



Contenido del nivel

Capítulo 1. Conceptos Fundamentales

- 1.1.- Dato e Información.
- 1.2.- Archivos.
- 1.3.- Base de Datos.
- 1.4.- Manejador de Base Datos.

Capítulo 2. Conceptos de Bases de Datos

- 2.1.- Campo.
- 2.2.- Tabla.
- 2.3.- Registro.

Capítulo 3. Crear Bases de Datos Sqlite

- 3.1.- Sqlite.
- 3.2.- Crear Bases de Datos.
- 3.3.- Manipular Los Datos.

Capítulo 4. Elementos de Bases de Datos

- 4.1.- Introducción.
- 4.2.- Formularios.
- 4.3.- Reportes y Consultas.
- 4.4.- Otros Elementos.

Capítulo 5. Formularios Para Tablas Simples

- 5.1.- Listados.
- 5.2.- Elementos de Captura de Datos.
- 5.3.- Registro y Edición.

Capítulo 6. Claves Únicas

- 6.1.- Índices.
- 6.2.- Clave Única o Candidata.
- 6.3.- Clave Candidata Compuesta.

Capítulo 7. Clave Primaria

- 7.1.- Clave Primaria.
- 7.2.- Campos Auto Numéricos.

Capítulo 8. Clave Foránea

- 8.1.- Clave Foránea.
- 8.2.- Fk y Campos Auto Numéricos.
- 8.3.- Clave Primaria Compuesta.

Capítulo 9. Datos de Claves Foráneas

- 9.1.- Datos Clasificadores.
- 9.2.- Datos Para Registros Múltiples.
- 9.3.- Datos Con Múltiples Claves Foráneas.

Capítulo 10. Integridad Referencial

- 10.1.- Integridad Referencial.
- 10.2.- Establecer Claves Foráneas.
- 10.3.- Validación de Los Datos.

Capítulo 11. Restricciones de Clave Foránea

- 11.1.- Desencadenantes.
- 11.2.- On Delete.
- 11.3.- On Update.

Capítulo 12. Interfaces de Usuario Con Clave Foránea

- 12.1.- Interfaces Con Clasificadores.
- 12.2.- Interfaces Con Registros Múltiples.
- 12.3.- Registros Múltiples Con Varias FK.

Capítulo 13. Relaciones Entre Tablas

- 13.1.- Bases de Datos Relacionales.
- 13.2.- Relación Uno a Uno.
- 13.3.- Relación Uno a Muchos.

Capítulo 14. Modelo Entidad Relación

- 14.1.- Concepto.
- 14.2.- Elementos.
- 14.3.- Símbolos.



Capítulo 1. CONCEPTOS FUNDAMENTALES

1.1.- Dato e Información

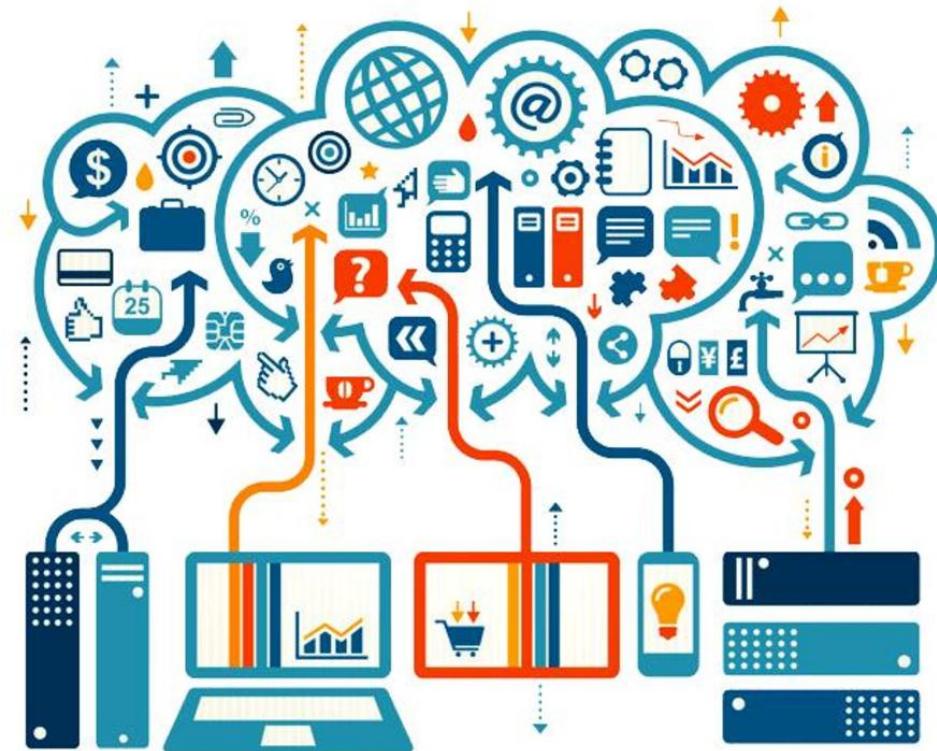
Un dato es un valor numérico o alfanumérico que es relevante en un contexto, por ejemplo, una organización, un evento o una entidad. Por ejemplo, en el contexto de una organización como una universidad un dato relevante es el nombre de una carrera; en una empresa, el monto de una factura es un dato relevante; en un juego de fútbol, el nombre del jugador que hizo gol es un dato relevante.

Generalmente en estos contextos se generan muchos datos relevantes que deben ser almacenados para usarlos en el futuro. En la siguiente imagen se muestra una serie de datos que corresponden a los alumnos que han presentado exámenes en un lapso de tiempo:

7514	12030	357	11690	2016-02-20	18:00:00	42	8	18:26:08	26	11
7515	13572	365	11700	2016-02-22	9:23:00	44	6	9:36:36	13	1
7516	14616	365	11702	2016-02-22	9:16:00	39	11	9:29:53	13	1
7517	13275	365	11704	2016-02-22	9:21:00	48	2	9:36:23	15	1
7518	14177	365	11710	2016-02-22	9:16:00	41	9	9:37:14	21	1
7519	13276	365	11703	2016-02-22	9:42:00	49	1	10:05:02	23	1
7520	14967	391	11726	2016-02-23	14:57:00	48	2	15:16:28	19	92
7521	8181	391	11720	2016-02-23	15:23:00	43	7	15:43:12	20	92
7522	14908	391	11723	2016-02-23	14:42:00	48	2	15:09:08	27	92
7523	5060	391	11728	2016-02-23	15:09:00	47	3	15:38:08	29	92
7524	14890	391	11724	2016-02-23	15:07:00	46	4	15:36:59	29	92
7525	14174	391	11725	2016-02-23	14:46:00	44	6	15:16:00	29	92
7526	14238	391	11727	2016-02-23	14:41:00	45	5	15:10:20	29	92
7527	13572	327	11738	2016-02-24	9:01:00	33	17	9:14:23	13	24
7528	14177	327	11747	2016-02-24	9:03:00	35	15	9:18:34	15	24
7529	14616	327	11743	2016-02-24	9:01:00	29	21	9:18:43	17	24
7530	13275	327	11745	2016-02-24	9:02:00	42	8	9:19:11	17	24
7531	13276	327	11744	2016-02-24	9:02:00	39	11	9:22:11	20	24
7532	13572	392	11749	2016-02-24	9:33:00	34	16	9:45:21	12	24
7533	12298	164	11732	2016-02-24	9:55:00	40	10	10:15:57	20	94
7534	14616	392	11755	2016-02-24	9:31:00	40	10	9:45:56	14	24
7535	14858	164	11737	2016-02-24	9:26:00	43	7	9:48:45	22	94
7536	15080	164	11731	2016-02-24	9:33:00	37	13	9:56:09	23	94
7537	13275	392	11756	2016-02-24	9:32:00	41	9	9:47:43	15	24

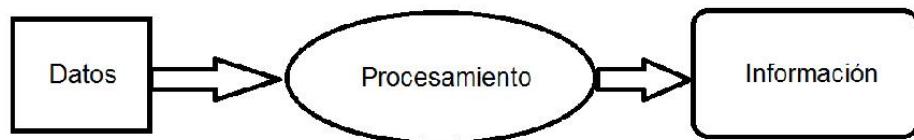
Big data o Macro datos (datos masivos) es un término que hace referencia al concepto relativo a conjuntos de datos tan grandes y complejos como para que hagan falta aplicaciones informáticas no tradicionales de procesamiento de datos para tratarlos adecuadamente. Lo que califica como "big data" varía según las

capacidades de los usuarios y sus herramientas, y las capacidades de expansión hacen que big data sea un objetivo en movimiento.



Los datos aisladamente pueden no contener información humanamente relevante. Solo cuando un conjunto de datos se examina y se procesa conjuntamente a la luz de un enfoque, hipótesis o teoría se puede apreciar la información contenida en dichos datos.

La información es un conjunto organizado de datos procesados, que constituyen un mensaje que cambia el estado de conocimiento del sujeto o sistema que recibe dicho mensaje.



Se considera que la generación y/u obtención de información persigue estos objetivos:

- Aumentar/mejorar el conocimiento del usuario, o dicho de otra manera reducir la incertidumbre existente sobre un conjunto de alternativas lógicamente posibles.
- Proporcionar a quien toma decisiones la materia prima fundamental para el desarrollo de soluciones y la elección.
- Proporcionar una serie de reglas de evaluación y reglas de decisión para fines de control.

1.2.- Archivos

Un archivo (o fichero) es una colección de información almacenada como una unidad en alguna parte de algún dispositivo de almacenamiento secundario de la computadora (disco duro, memoria extraíble, cd, dvd, etc), identificado por un nombre, una extensión (opcional) y la descripción de la carpeta o directorio que lo contiene.

Esta colección de datos sirve para entrada y/o salida a la computadora y se maneja con un programa específico que sabe interpretar el contenido del archivo. En contraste con una variable de programas, no tiene un tamaño fijo y está limitado solo por el espacio disponible en el dispositivo donde se está almacenando, es decir, su tamaño es dinámico.



La necesidad de un archivo está dada justamente por la diferencia que existe entre la memoria RAM y los medios de almacenamiento donde se guardan los archivos: la memoria RAM es volátil (es decir, su contenido se pierde al cerrar un programa o apagar la computadora) y un disco duro (o similar) no lo es. Por lo tanto, si todos los datos introducidos por el usuario al programa, se almacenan solo en variables, estos datos se perderán al cerrar el programa.

El tamaño de los archivos no está limitado por la memoria RAM de la computadora, está limitado solo por el tamaño de la unidad de almacenamiento o por el espacio disponible en ella. Aunque en la práctica, cuando abrimos un archivo, debe cargarse en la memoria RAM para poder ser modificado (esto es transparente para el programador), por lo que para abrir un archivo grande necesitamos tener suficiente memoria para poder manipular su contenido.

Uno de los tipos de archivos más usados para almacenar datos son los llamados archivos de texto plano, aunque también pueden usarse otros formatos, como el de MS Excel.



El uso de archivos para almacenar datos que serán usados por un sistema tiene algunas desventajas, entre ellas podemos mencionar:

- Conurrencia: los datos pueden ser accedidos y modificados solamente por una persona a la vez.
- Integridad: si más de una aplicación usa los datos del archivo, puede corromperse.
- Relación entre datos: es muy difícil establecer relaciones entre los datos.
- Seguridad: los datos pueden ser accedidos por cualquier usuario.

1.3.- Base de Datos

Es una colección organizada de datos relacionados. Se utiliza para almacenar y recuperar datos. Físicamente los datos se almacenan como archivos. La diferencia con los archivos, es que la base de datos almacena los datos de una forma tal que pueden relacionarse entre ellos y es más fácilmente manejable que los archivos trabajados aisladamente, agregando así algo más de valor.

Hay 3 palabras claves en las bases de datos:

- organizado: hace referencia a la manera en que los datos deben ser almacenados.
- colección: los datos se agrupan aunque sean de distintos tipos.
- inter-relacionado: interrelación entre los datos que permite que una colección de datos proporcione información razonable y coherente.

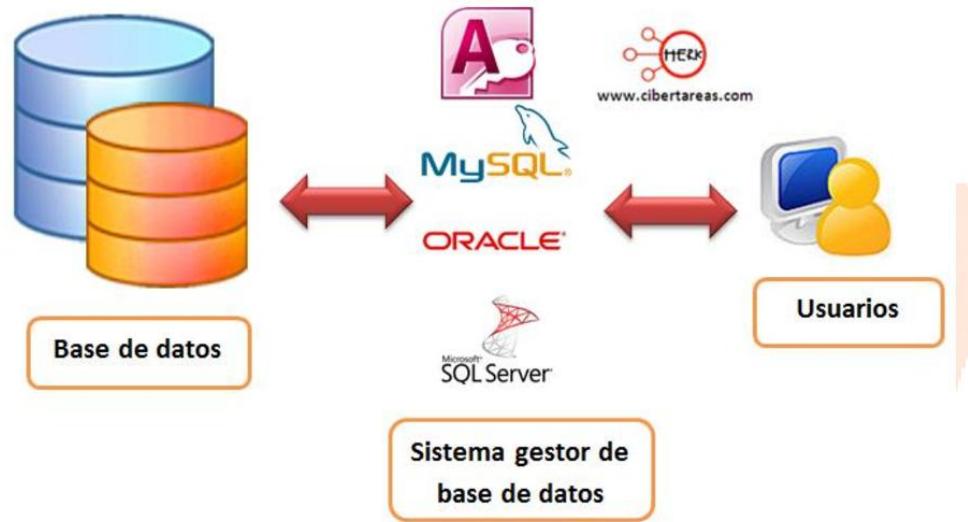


Existe un modelo de base de datos llamado "base de datos relacional", donde los datos están relacionados entre ellos. Este modelo es uno de los más usados en el desarrollo de sistemas de información y aplicaciones en general. El modelo relacional tiene la ventaja de soportar un modelo matemático formal, que basa en el álgebra relacional y el cálculo relacional. Otra ventaja de este modelo es que tiene capacidad de hacer cumplir las restricciones de integridad de datos.



1.4.- Manejador de Base Datos

Los Sistemas de Gestión de Bases de Datos (SGBD, en inglés Data Base Management System, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.



Existen distintos objetivos que deben cumplir los SGBD:

- Abstracción de la información. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos.
- Independencia. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- Seguridad. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura de permisos a usuarios.
- Manejo de transacciones. Una transacción es un programa que se ejecuta como una sola operación, esto quiere decir que luego de una ejecución en la que se produce una falla es el mismo que se obtendría si el programa no se hubiera ejecutado.

Ventajas de los manejadores de bases de datos:

- Proveen facilidades para la manipulación de grandes volúmenes de datos
- Simplifican la programación de equipos de consistencia.

- Manejando las políticas de respaldo adecuadas, garantizan que los cambios de la base serán siempre consistentes sin importar si hay errores correctamente, etc.
- Organizan los datos con un impacto mínimo en el código de los programas.
- Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos

Entre los manejadores de bases de datos más usados están:

- MySQL Licencia Dual, depende el uso
- PostgreSQL Licencia BSD
- SQLite
- Firebird basada en la versión 6 de InterBase
- dBase
- Fox Pro
- IBM DB2 Universal Database (DB2 UDB)
- IBM Informix
- Interbase de CodeGear,
- Microsoft SQL Server
- Open Access
- Oracle
- Progress (DBMS)



Capítulo 2. CONCEPTOS DE BASES DE DATOS

2.1.- Campo

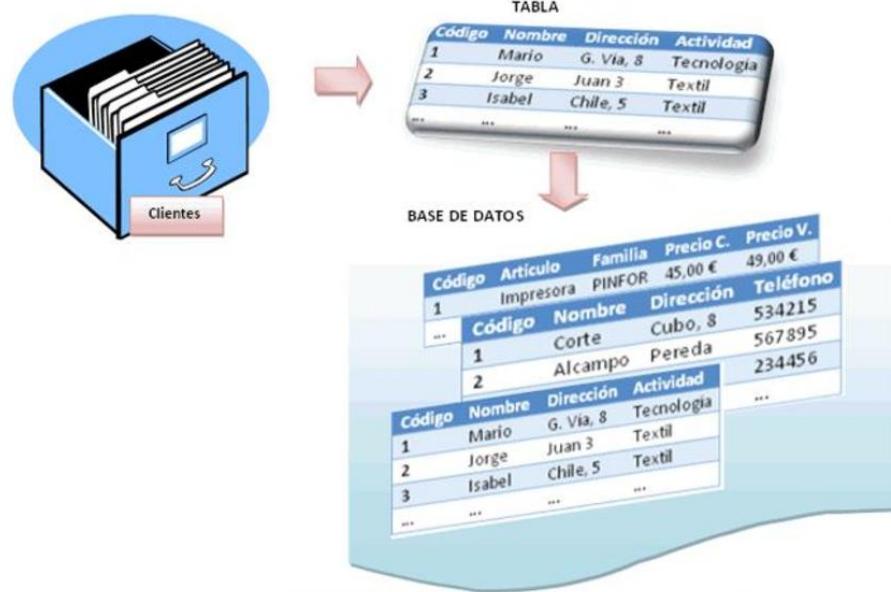
Un campo es la unidad mínima en una base de datos. Es un dato sobre el cual se necesita almacenar información, por ejemplo: nombre, teléfono, dirección, etc. Cada campo debe tener un nombre y un tipo de dato, que va a depender de la naturaleza del dato mismo (como las variables en la programación), por ejemplo: el campo "cédula" debe ser alfanumérico, el campo "peso" debe ser entero o real, etc. El tipo de dato específico dependerá del manejador de base de datos, aunque típicamente hay tipos carácter, numérico, fechas, entre otros.

Se recomienda que los nombres de los campos sean significativos, sin espacios en blanco y sin caracteres especiales que puedan entorpecer posteriormente la programación. La cantidad de campos dependerá de cada caso.

2.2.- Tabla

Una tabla es un conjunto de campos que guardan una relación entre si y almacenan datos sobre una entidad o elemento de importancia para un sistema, por ejemplo: clientes, proveedores, productos, ventas, etc. Son los objetos principales de bases de datos que se utilizan para guardar datos. Cada tabla creada debe tener un nombre único en la base de datos. En una tabla no puede haber 2 campos con el mismo nombre.

En las bases de datos relaciones, las tablas se relacionan entre sí a través de algunos campos. Una base de datos puede tener tantas tablas como sea necesario. La cantidad de tablas es relativa a los requerimientos.



2.3.- Registro

Un registro es una colección o agrupación de valores de los campos. Por ejemplo, si tenemos los campos cédula, nombre y teléfono, un registro sería los valores de esos campos de una persona: 14978150, José Rojas, 04165194596. A los registros también se les llama "fila de una tabla", como se muestra en la siguiente imagen:

Campos

Registros

Nombre Proyecto	Contacto	Correo
Eagle	anell y adrian	eagleog@cantv.net
TLC	Ambar Texeira	boutiquepcuentas2003@gmail.com
Tedexis	Luis Poggi	lpoggi@gmail.com
FYC	Carlos Marrero	carlos.marrero@fyccorp.com
NGS camcareni	Alfonso Mora	amora@ngs.com.ve
Tecnoquim	Lenin Marques	grupoteonoquim@gmail.com
Tuscomprasporinternet	Rafael Monserrat	bufetemontserrat@hotmail.com

En resumen, en las bases de datos se deben dar las siguientes características:

- una tabla tiene múltiples columnas.

- cada columna tiene un nombre único y contiene conjuntos de datos.
- una tabla es una colección de registros de una base de datos.
- cada conjunto de datos se denomina fila (registro).
- un valor se obtiene por la intersección de una fila y de una columna.



Capítulo 3. CREAR BASES DE DATOS SQLITE

3.1.- Sqlite

SQLite es una librería de lenguaje C que implementa una pequeña, rápida y auto contenido motor de base de datos SQL. Es multiplataforma y altamente compatible con la mayoría de los lenguajes de programación. Con esta librería se crea una base de datos es un simple archivo que contendrá toda la base de datos. El código fuente de SQLite es de dominio público y está libre para ser usado para cualquier propósito. Se puede descargar de la página oficial <https://sqlite.org/index.html>.

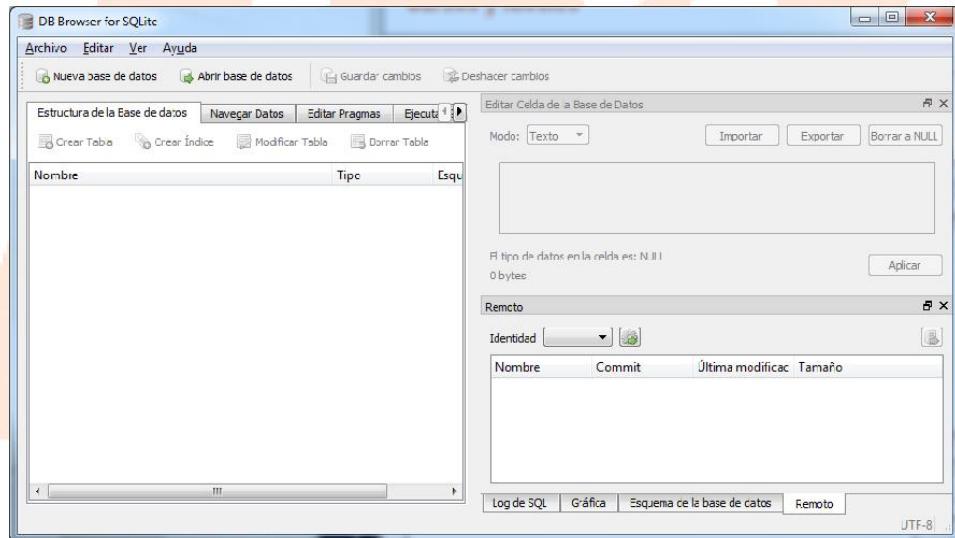
De la página oficial se descarga la librería y algunas herramientas de línea de comandos, pero no se ofrece una interfaz gráfica para trabajar con las bases de datos de una forma simple y amigable.



"DB Browser for SQLite" o DB4S es una herramienta visual, open source (código abierto) para crear, diseñar y editar bases de datos de SQLite. La última versión puede ser descargada del sitio web oficial <https://sqlitebrowser.org/>. Existe una versión para Windows, para Linux y para MacOs.

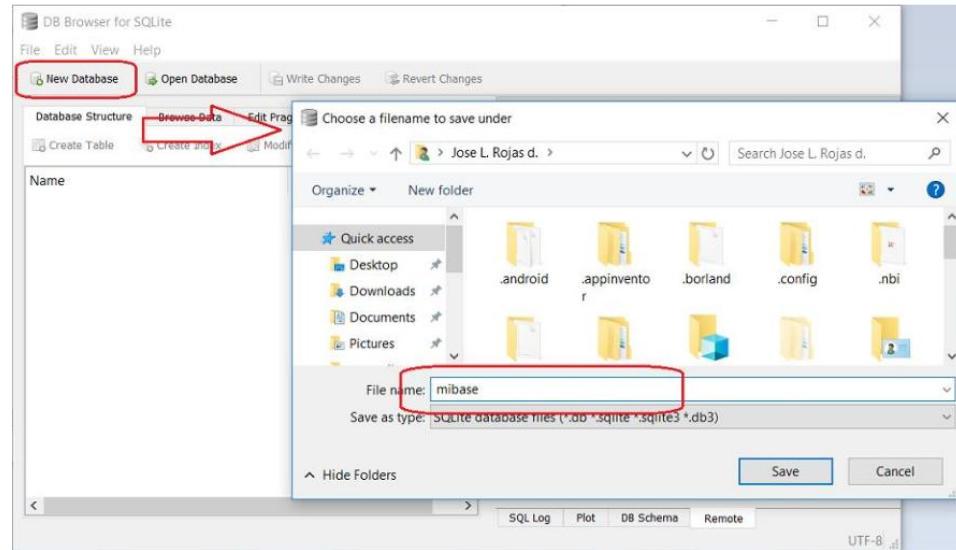


Luego de instalada la herramienta, al abrirla se muestra una ventana como la siguiente, donde se puede crear una base de datos nueva o abrir una base de datos existente:

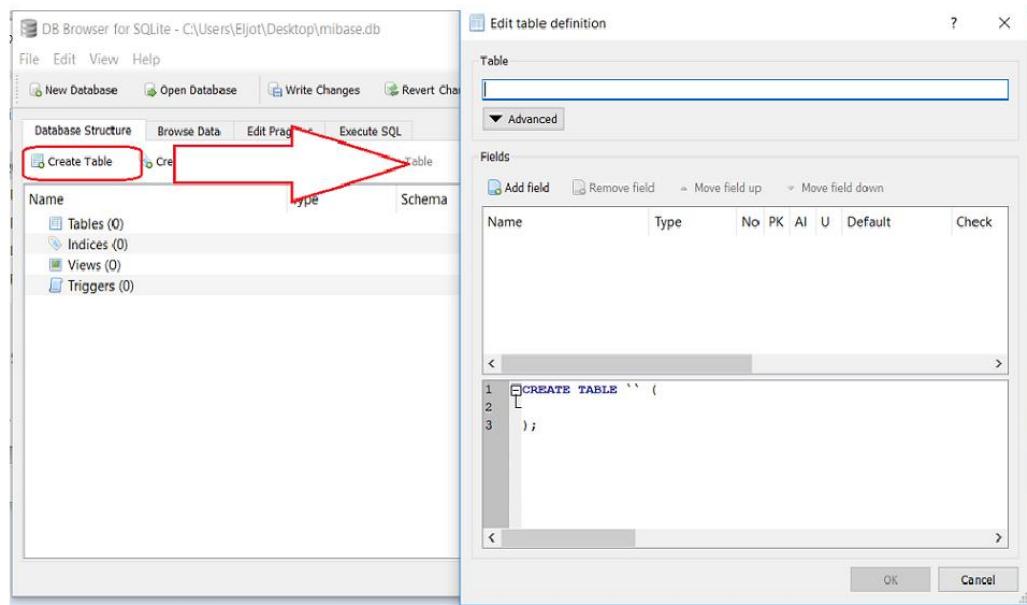


3.2.- Crear Bases de Datos

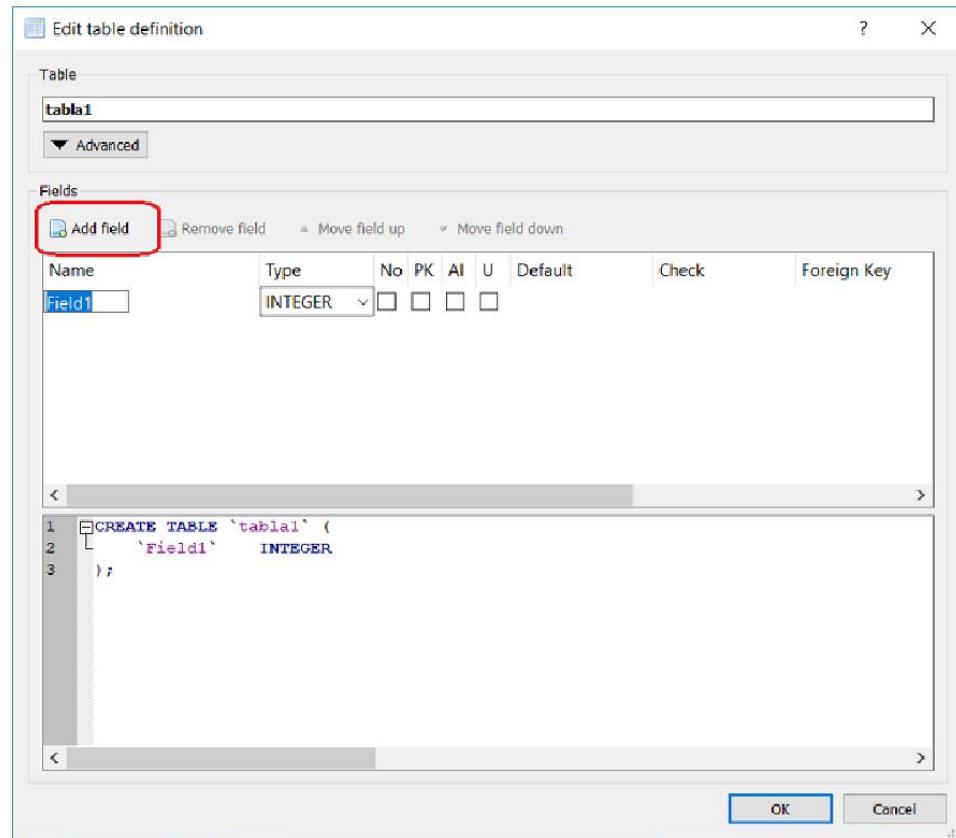
Para crear una base de datos usando DB Browser for SQLite se hace click en el botón "New Database" que se encuentra en la barra de herramientas. Luego debe indicarse la ruta donde se guardará la base de datos:



Luego de crear la base de datos se abre automáticamente la ventana para crear una nueva tabla. También se puede hacer click en el botón "Create Table":



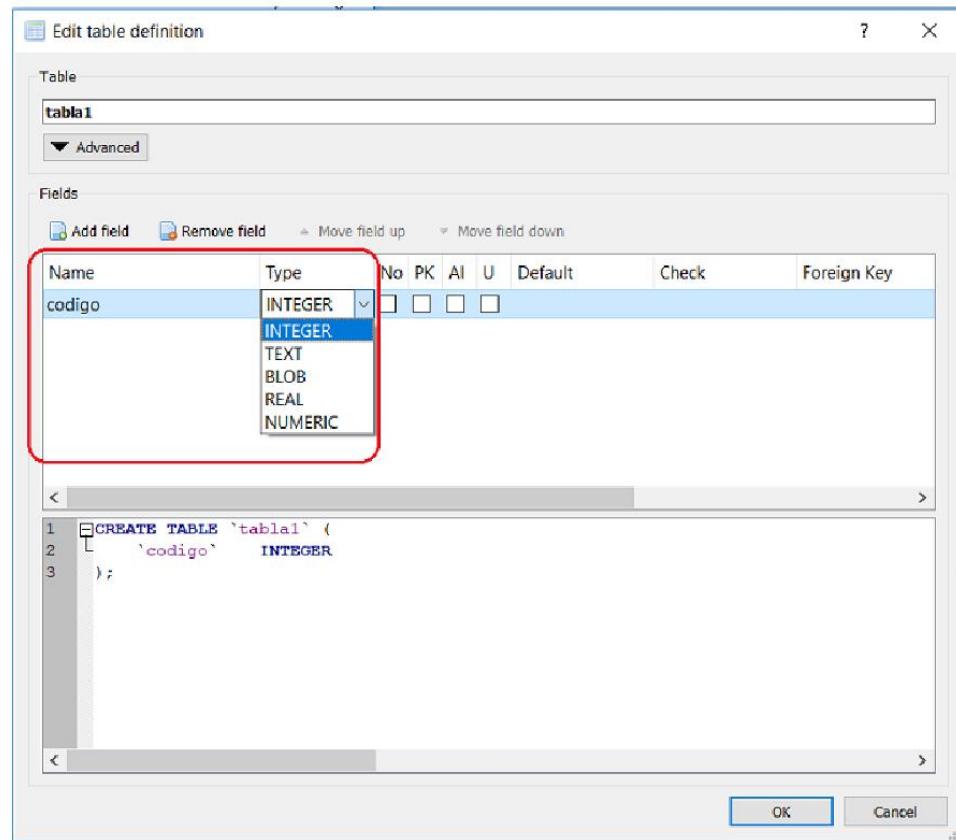
Al crear la tabla se debe indicar su nombre y los campos que ésta contendrá. Se debe hacer click en el botón "Add Field" tantas veces como campos se desean definir en la tabla, como se muestra en la imagen:



Cada campo debe tener un nombre y un tipo de dato. Cada manejador de bases de datos tiene los tipos de datos particulares. SQLite sólo posee:

- Integer: para datos numéricos enteros.
- Text: para datos alfanuméricos.
- Blob: para datos de tipo archivos, por ejemplo fotos.
- Real: para datos numéricos reales (números con decimales).

Para SQLite los tipos de datos de los campos no restringen los posibles valores que pueden almacenarse en ellos porque finalmente todo se maneja en texto. Esto puede ser confuso porque en la mayoría de DBMS el tipo de dato del campo restringe los valores que puede recibir.

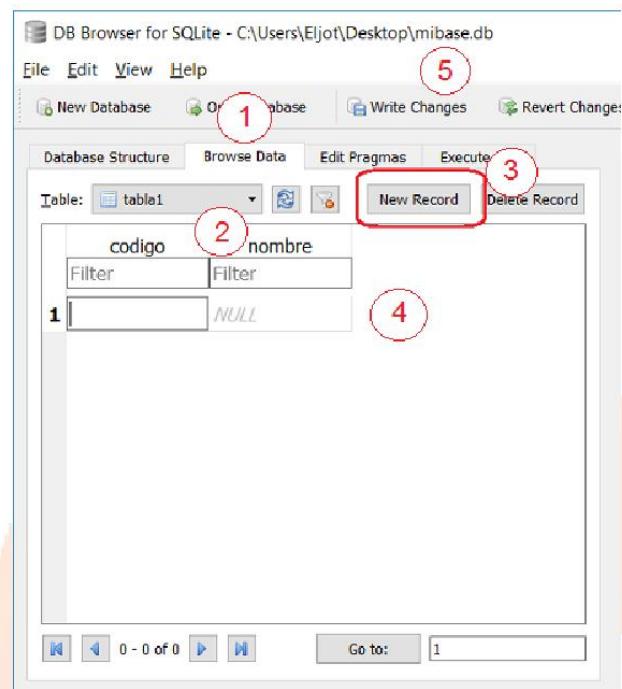


3.3.- Manipular Los Datos

Luego de creada la tabla se pueden agregar, modificar o eliminar registros. Para agregar registros se deben realizar los siguientes pasos:

- 1- Hacer click en la pestaña "Browse data"
- 2- Seleccionar la tabla a la cual se desea manipular los datos.
- 3- Hacer click en el botón "New Record".
- 4- Escribir los valores en la nueva fila que se genera.
- 5- Luego de agregar todos los registros, se debe hacer click en el botón "Write Changes".

Los pasos 3 y 4 se deben repetir tantas veces como registros nuevos se desean agregar.



Capítulo 4. ELEMENTOS DE BASES DE DATOS

4.1.- Introducción

Al trabajar con bases de datos se presentan 3 momentos:

- 1- el diseño de la base de datos: también llamado modelado de los datos. Lo realiza el arquitecto de software independientemente del manejador de bases de datos, basándose en los requerimientos del sistema, en el modelo y las reglas de negocio.
- 2- la creación de la base de datos: usando un manejador de bases de datos específico. Lo realiza el DBA (Database Administrator) o un programador con conocimiento del manejador de bases de datos seleccionado.
- 3- la manipulación de los datos, que abarca 2 sub momentos:
 - a- las operaciones básicas CRUD: (Create, Read, Update y Delete o en español Registrar, Leer, Actualizar y Eliminar), que son llevadas a cabo por los usuarios (típicamente del nivel operativo) de la organización a través de las aplicaciones (o FrontEnds), que validan los datos y se aseguran que se cumplan las reglas de negocio.
 - b- el procesamiento de los datos: para obtener información relevante para tomar decisiones. También se hace con los FrontEnds, por lo general, por usuarios con nivel de responsabilidad mayor en la organización.

Por tal razón, al trabajar con bases de datos no solamente se trata con las tablas y los datos (registros), también hacen falta otros elementos, tales como:

- formularios.
- informes o reportes.
- vistas o consultas.
- procedimientos almacenados.
- triggers o disparadores.

Algunos manejadores de bases de datos permiten incluir dentro de la misma base de datos todos estos elementos; otros sólo permiten algunos, por ejemplo, Oracle los permite crear todos, pero Mysql no permite crear formularios. Aunque el manejador de bases de datos permita crear todos los elementos, normalmente estos se materializan (se crean) en las aplicaciones o FrontEnds y se deja la base de datos solo para almacenar datos. La selección del manejador de base de datos puede estar

influido por esto y la decisión de cuales elementos crear en la base de datos y cuales en los FrontEnd es muy arbitraria.

4.2.- Formularios

Generalmente en una organización trabajan varias personas que estan generando datos o que necesitan consultar datos, por ejemplo, un cajero de un banco que registra el retiro de dinero de un cliente, o un ejecutivo que consulta el estatus de alguna solicitud realizada previamente por el cliente. Además, la empresa debe asegurarse que se cumplan las reglas de negocio, por ejemplo, no se puede retirar de una cuenta una cantidad de dinero mayor que el saldo disponible, no se puede modificar el saldo de una cuenta, ciertos usuarios no pueden hacer algunas actividades pero otros si, etc.

Es por esta razón, que la manipulación de los datos de una base de datos rara vez se realiza directamente en el manejador de base de datos, editando los datos en las tablas. Cuando se hace, lo debería hacer un usuario con privilegios y con conocimiento de bases de datos para reducir la posibilidad de cometer errores al manipular los datos. Normalmente este trabajo es realizado por un DBA o un programador experimentado al cuál se le han delegado actividades de DBA.

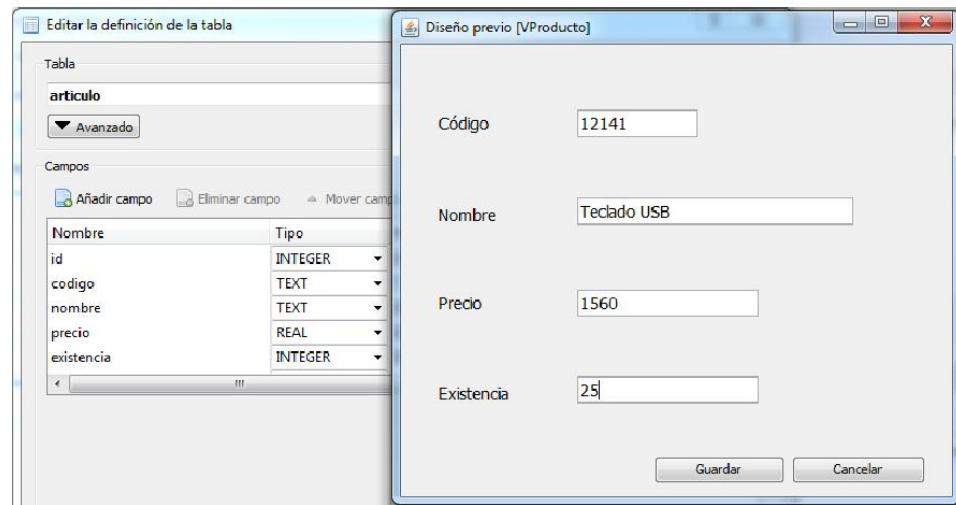
Entonces, se hace necesario un mecanismo para la manipulación de los datos en el día a día de una organización por parte de sus integrantes, de forma tal que se logren manipular los datos, los datos sean válidos y que a la vez se cumplan las reglas del negocio.

En el argot del desarrollo de software, se le llama "formulario" a una ventana o interfaz de usuario, que le permite a éste interactuar con los datos de una o más tablas de una base de datos. Esta interfaz contiene la programación necesaria para lograr los siguientes objetivos:

- validar los datos antes de guardarlos en la base de datos.
- que se cumplan las reglas del negocio.
- garantizar la seguridad de los datos.
- hacer fácil y transparente para el usuario la manipulación de los datos.

Con los formularios típicamente se realizan las operaciones CRUD, aunque según las reglas de negocio, en algunas tablas no se deben permitir ciertas operaciones, por ejemplo eliminar registros, o ciertos datos no se pueden modificar.

Una interfaz de usuario puede manipular o usar una o varias tablas de la base de datos. Los elementos que debe poseer una interfaz de usuario deben ser coherentes con la(s) tabla(s) que va a manipular, por ejemplo en una tabla sencilla donde se guardan los datos de los clientes con los siguientes campos, se puede tener un formulario como el siguiente:



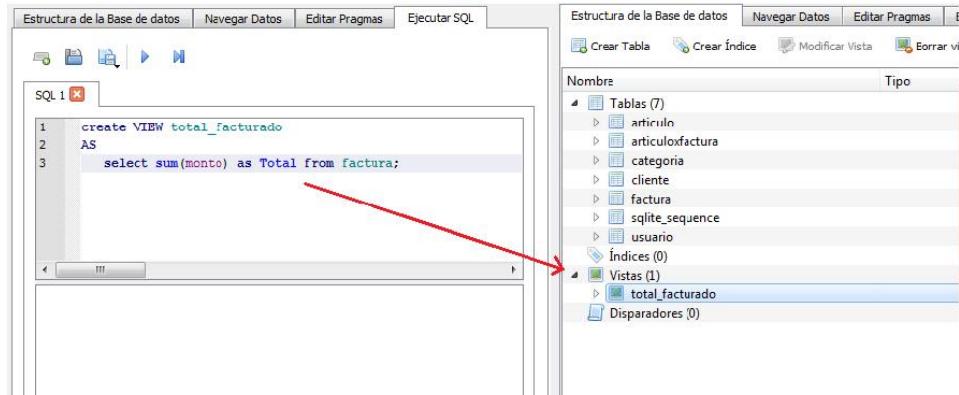
4.3.- Reportes y Consultas

Al tener los datos almacenados en la tablas, en cualquier momento se puede requerir consultar estos datos, ya sea buscando un registro en particular o haciendo resúmenes de los datos utilizando fórmulas matemáticas o estadísticas, por ejemplo, contar cuantos registros contienen cierto valor en alguno de sus campos o sumando los valores de ciertos campos de ciertos registros, etc. A estas operaciones se les llama consultas, porque literalmente se están consultando los datos almacenados en las tablas.

Para hacer estas consultas se utiliza un lenguaje estandard de consultas llamado: Structure Query Language (lenguaje de consultas estructurado), conocido por las siglas SQL. Estas consultas son ejecutadas por el manejador de base de datos. Se pueden ejecutar al vuelo para probarlas, se pueden ejecutar desde los FrontEnds o BackEnd o se pueden guardar en la base de datos para ser usadas posteriormente. Cuando se almacenan en la base de datos se les llama "vistas" o "view" en inglés.

Las vistas no almacenan datos, solo consultan los datos de una o varias tablas y generan en tiempo de ejecución una lista de registros que resultan de la ejecución de la consulta. En el segundo nivel del curso de estudiaron a fondos las instrucciones SQL.

En la siguiente imagen se muestra un ejemplo de cómo crear una vista en SQLite:



Los reportes o informes también son interfaces, pero creadas específicamente para ser impresas, por ejemplo: presupuestos, facturas, ingresos del mes, clientes morosos, etc. Los reportes generalmente utilizan datos en bruto de varias tablas y también muestran información procedente del procesamiento de los datos.

En la siguiente imagen se muestra una serie de gráficas y valores resultantes del procesamiento de datos de base de datos.



4.4.- Otros Elementos

Además de los datos y las consultas, en las bases de datos también puede haber código de programación para manipular o procesar los datos almacenados. El código puede almacenarse en:

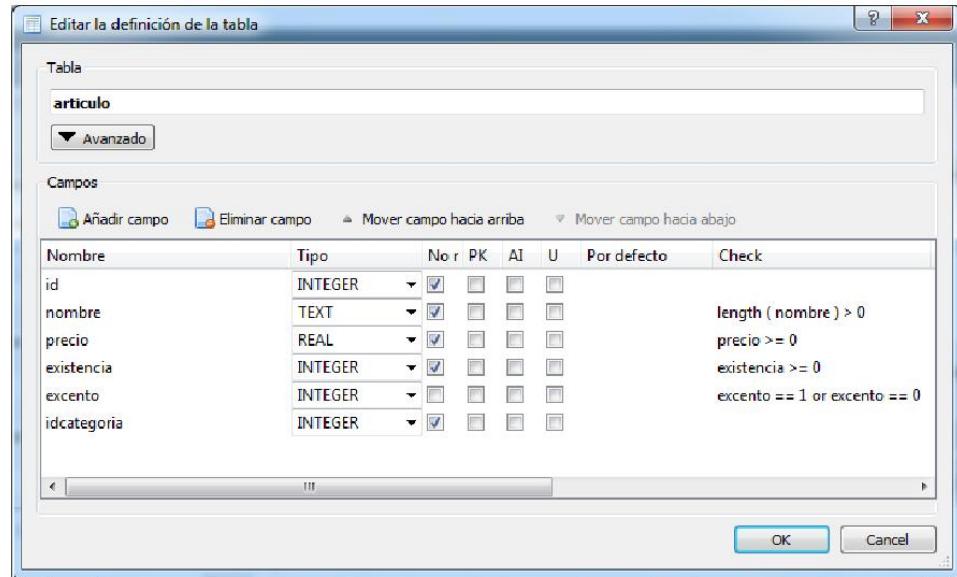
- restricciones o validaciones de campos y/o de tablas (constraints).
- procedimientos almacenados (stored procedures).
- disparadores (triggers).

Esta estrategia tiene 2 ventajas:

- mantener el código en la base de datos para no tener que repetirlo en cada FrontEnd.
- lograr que la ejecución sea más eficiente, porque al tener el código junto con los datos, es más rápido y fácil el acceso.

Cada manejador de base de datos tiene su propio lenguaje de programación para crear estos elementos. Sqlite permite crear triggers y restricciones (tanto de campo y

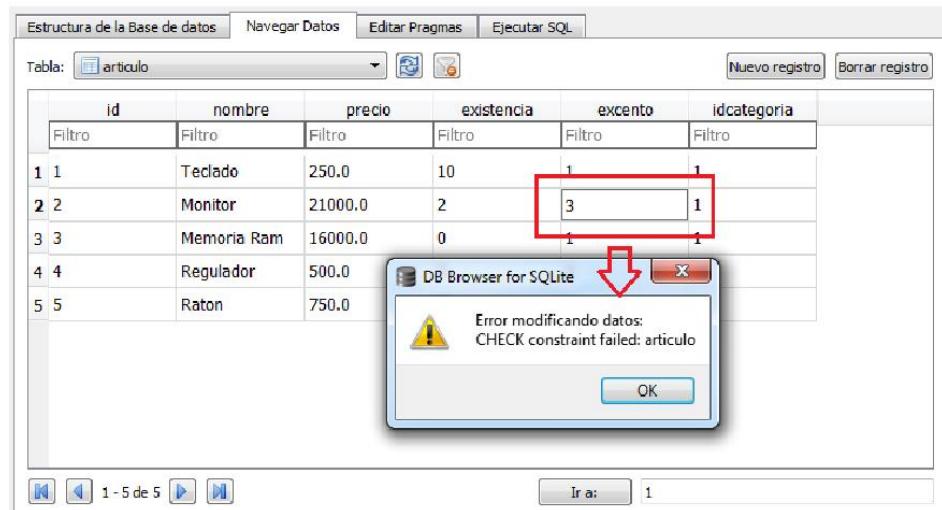
de tabla), pero no permite crear procedimientos almacenados. En la siguiente imagen se muestra como agregar una restricción de campo en Sqlite:



En "DB Browser for Sqlite", en la columna "check" de cada campo de la ventana "modificar tabla" se escriben las evaluaciones lógicas que se deben cumplir para considerar que el valor del campo es válido o no. Esta validación se ejecuta al momento de que un campo reciba un valor, ya sea en la interfaz de la herramienta o través de un FrontEnd.

Como se ve en el ejemplo, se colocaron en algunos campos expresiones lógicas que hacen referencia al nombre del campo. Si el valor que se le esta intentando asignar a un campo no hace que se cumpla la expresión lógica, se genera un error a nivel de la base de datos y el valor no se guarda.

En la siguiente imagen se muestra el error que se genera al intentar guardar en un campo un valor que no cumple con la regla de validación:



Capítulo 5. FORMULARIOS PARA TABLAS SIMPLES

5.1.- Listados

Para realizar las operaciones CRUD generalmente se utiliza una interfaz que primero permite hacer la operación Read (leer o consultar). Para esto, típicamente se utiliza un elemento que permite mostrar al usuario un listado de los valores que existen en la tabla. Por ejemplo:

Name	Age	Nickname	Employee
Giacomo Giulizoni Founder & CEO	40	Peldi	
Marco Botton Tuttofare	38		
Mariah MacLachlan Better Half	41	Potata	
Valerie Liberty Head Chef	25	Val	

En los listados, típicamente no se muestran todos los campos de la tabla. Se muestran los campos más relevantes de la ésta. También es común que exista una forma de buscar o filtrar los registros utilizando alguno de los campos. Adicionalmente, previendo que el listado de registros sea muy largo, debe existir alguna forma de paginar los registros.

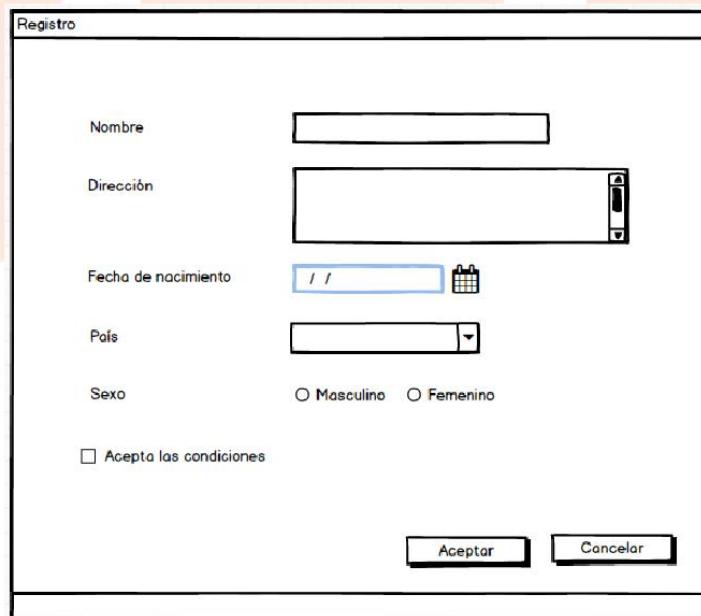
En las interfaces donde se muestran los listados no se acostumbra permitirle al usuario editar directamente los valores. Es por eso que hacen falta otras interfaces para las operaciones Create (registrar) y Update (modificar). Generalmente también se provee un mecanismo para hacer la operación Delete (eliminar), si la regla de negocio para esa tabla lo permite.

5.2.- Elementos de Captura de Datos

Existen varios elementos para la captura de datos que se pueden utilizar en las interfaces o formularios. Estos elementos deben ser coherentes con el dato que se va a capturar y con el tipo de dato del campo que lo va a almacenar. Entre los elementos están:

- campos de texto (textinput): utilizado para leer valores numéricos o valores de texto corto, por ejemplo: nombre, cédula, teléfono, etc.
- áreas de texto (textarea): utilizado para leer textos largos, por ejemplo: dirección, descripción de un producto, etc.
- checkboxes: utilizado para valores lógicos (verdadero o falso). Puede haber una lista de éstos, donde se pueden seleccionar varios porque son independientes, por ejemplo, acepta los términos y condiciones, etc.
- listas de selección (combobox): utilizado para mostrar al usuario una lista de valores de forma que el pueda seleccionar uno de éstos y no tenga la necesidad de escribirlos, por ejemplo: país de procedencia, ciudad, etc.
- botón de selección (radio button): utilizado para seleccionar un valor de varios posibles valores excluyentes entre éstos, por ejemplo: sexo, estado civil.
- selector de fecha (date chooser): utilizado para seleccionar fechas, por ejemplo, fecha de nacimiento, fecha de vencimiento, etc.

En la siguiente imagen se muestra el uso de cada uno de los elementos:



El formulario titulado "Registro" muestra los siguientes campos:

- Campo de texto para "Nombre".
- Campo de texto para "Dirección".
- Campo de texto para "Fecha de nacimiento" con un icono de calendario.
- Campo desplegable para "País".
- Botones de radio para "Sexo": "Masculino" y "Femenino".
- Un checkbox para "Acepta las condiciones".
- Botones "Aceptar" y "Cancelar" en el pie del formulario.

5.3.- Registro y Edición

En las interfaces para hacer Create (registrar) se deben mostrar todos los elementos necesarios para capturar los datos del usuario, excepto aquellos que sean calculados o que sean de uso directo para éste. Por ejemplo:

The diagram illustrates a web-based form for creating a new employee record. At the top, there is a header bar with standard browser controls (back, forward, stop, refresh) and a URL field showing "http://". Below the header, the main content area has a title "Información" and a sidebar containing a list of names. The main form contains four input fields: "Name", "Age", "Nickname", and "Employee". To the right of the form is a vertical sidebar with buttons for "Add", "Edit", and "Delete". At the bottom of the form are two buttons: "Ok" and "Cancel".

En las interfaces para hacer Create (registrar) típicamente también se realiza la operación Update (actualizar), con la diferencia de que los elementos de captura de datos se inicializan con los valores del registro seleccionado para ser editado.

Capítulo 6. CLAVES ÚNICAS

6.1.- Índices

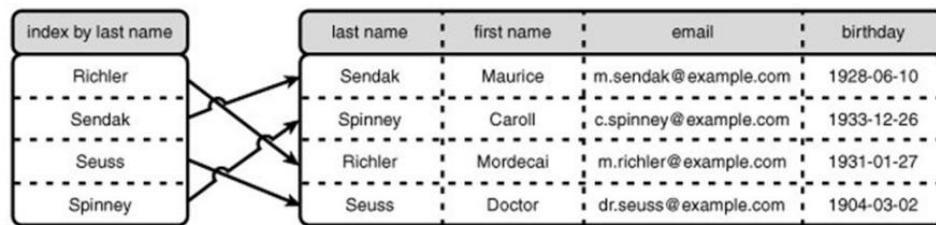
Los registros en las tablas se van agregando siempre al final de ésta, por lo tanto, siempre el registro más reciente va a estar ubicado al final de la tabla. El problema es que generalmente se requiere que los registros sean ordenados por criterios particulares, es decir, usando como referencia ciertos campos, por ejemplo: la fecha de nacimiento, el nombre, etc. Para lograr este objetivo sin necesidad de mover grandes cantidades de datos, los manejadores de base de datos utilizan índices.

El índice de una base de datos es una estructura de datos que mejora la velocidad de las operaciones, por medio de identificador de cada fila de una tabla, permitiendo un rápido acceso a los registros de una tabla en una base de datos.

El índice en una base de datos tiene un funcionamiento similar al índice de un libro, guardando parejas de elementos: el elemento que se desea indexar y su posición en la base de datos. Para buscar un elemento que esté indexado, sólo hay que buscar en el índice dicho elemento para, una vez encontrado, devolver un registro que se encuentre en la posición marcada por el índice.

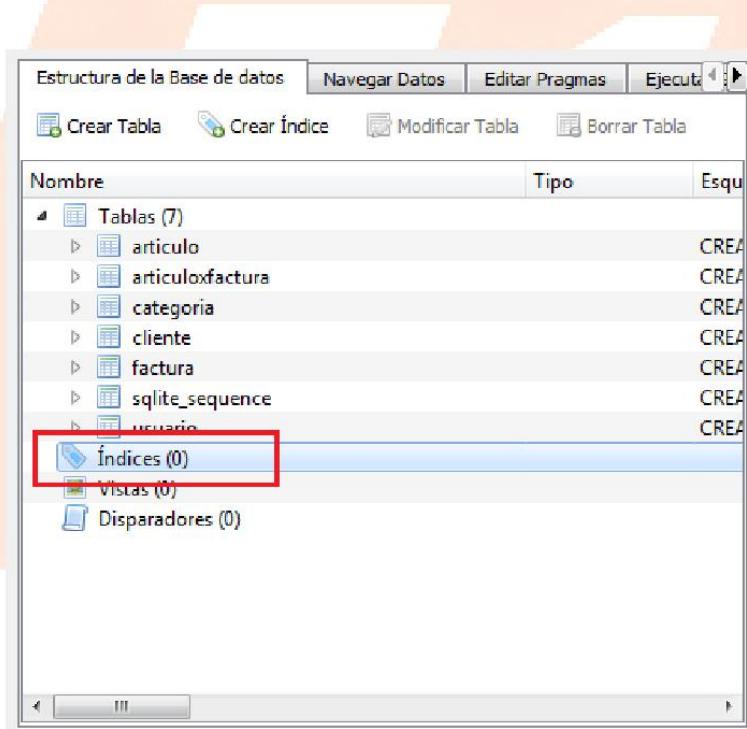
Los índices son opcionales, sin embargo, es una buena práctica definirlos al momento de crear las tablas. Una tabla puede tener uno o varios índices. Luego de definido el(los) índices, el manejador de base de datos se encarga de actualizarlos automáticamente y de forma transparente para el usuario final, inclusive para el DBA:

El espacio en disco requerido para almacenar el índice es típicamente menor que el espacio de almacenamiento de la tabla (puesto que los índices generalmente contienen solamente el(los) campos que se utilizaron para definirlo y excluyen el resto de los campos de la tabla), lo que da la posibilidad de almacenar en memoria los índices de tablas que no cabrían en ella.

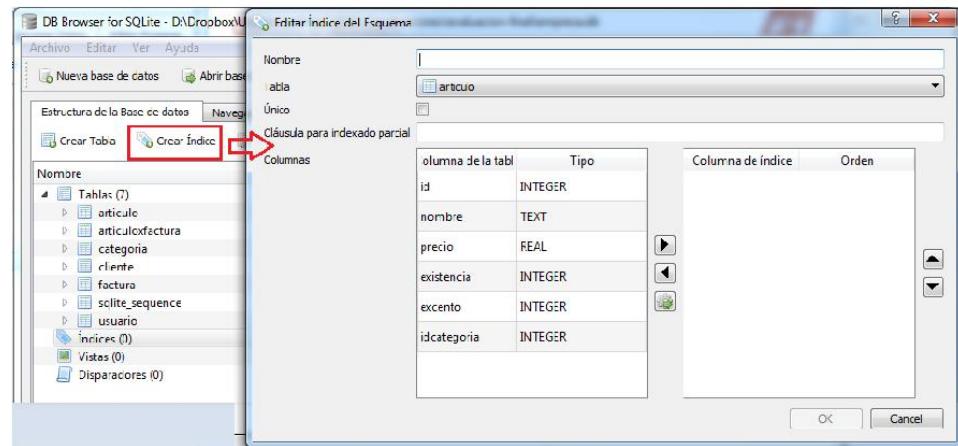


En la imagen anterior, se muestra una tabla con los campos: last name, first name, email y birthday, con 4 registros que fueron agregados cronológicamente. También se creó un índice usando el campo last name, donde están los valores de ese campo ordenados alfabéticamente, apuntando a su respectivo registro.

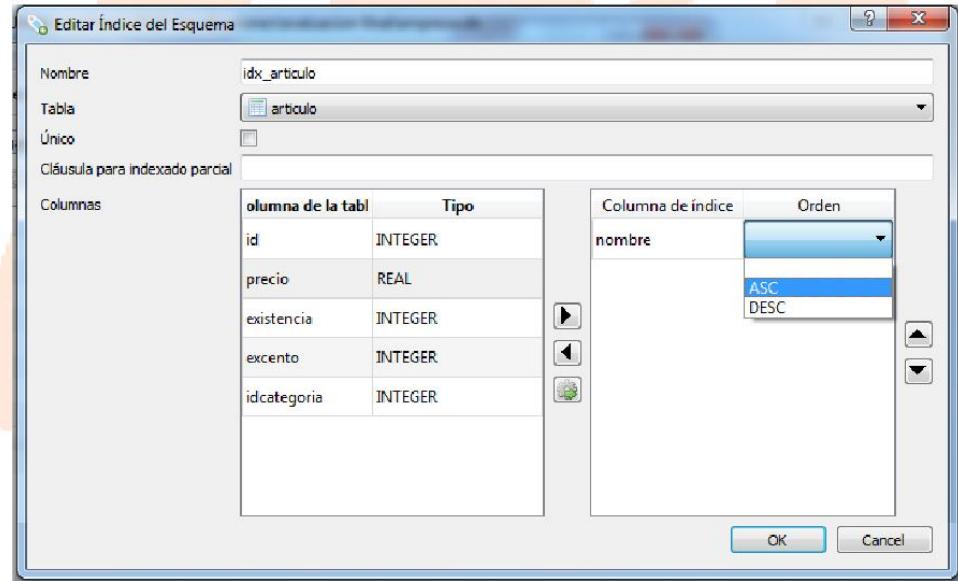
SQLite permite crear índices manualmente. La cantidad de índices existentes (que inicialmente es cero (0)), se muestra debajo de las tablas.



Al crear un índice se debe establecer un nombre (que no puede repetirse entre los nombres de los índices) y la tabla que indexará:



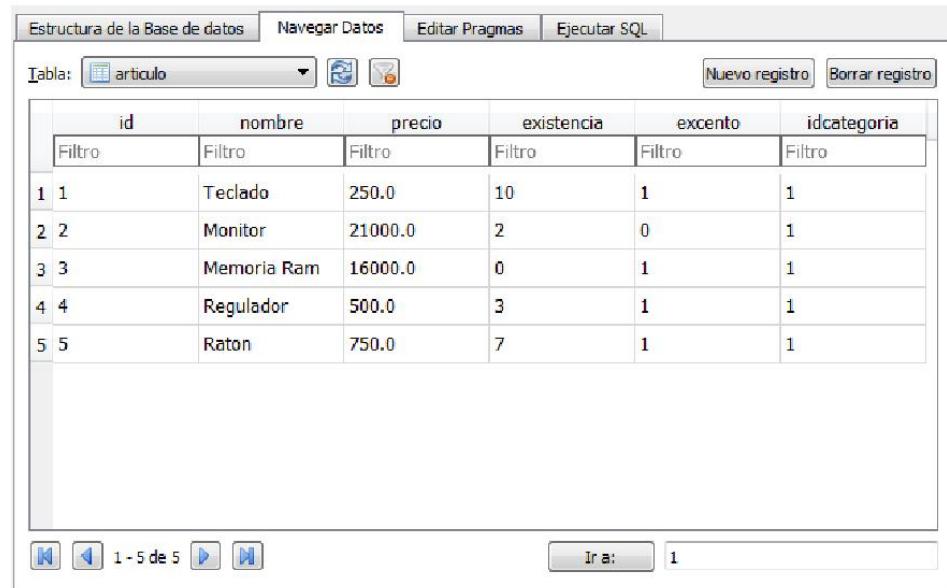
Se deben pasar hacia la derecha los campos que se usaran para indexar los registros y seleccionar el tipo de ordenamiento que se va a usar, que puede ser ascendente o descendente:



6.2.- Clave Única o Candidata

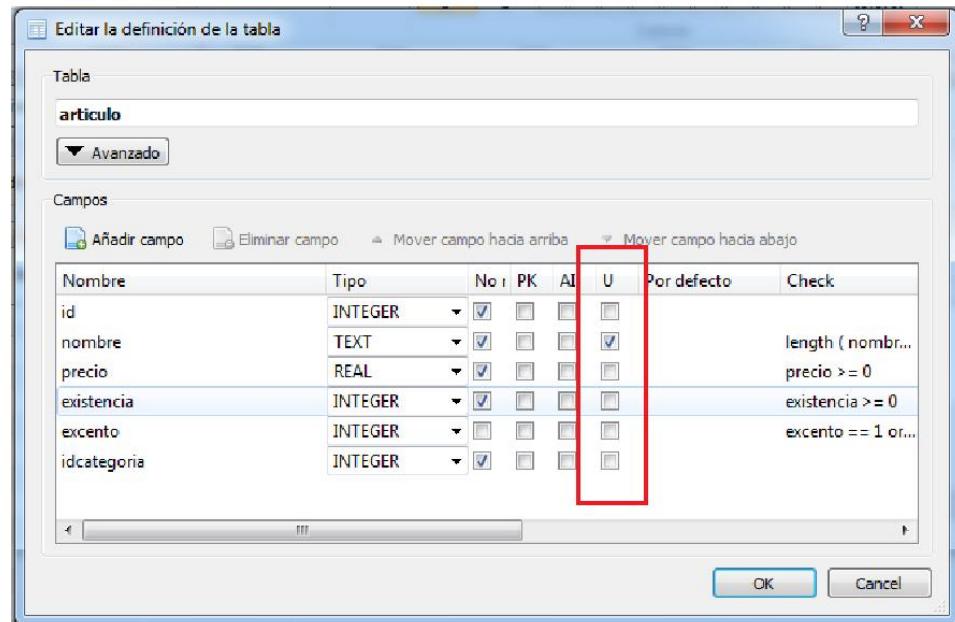
Entre todos los campos de una tabla, existen campos cuyos valores se pueden repetir entre los registros y otros que no se pueden repetir. Una clave candidata o clave

única es un campo cuyo valor no se debe repetir entre los registros de una tabla. En el siguiente ejemplo, una tabla donde se almacena los datos de los artículos que vende una empresa, el campo "nombre" es un valor único, puesto que no se debe registrar 2 veces el mismo producto:

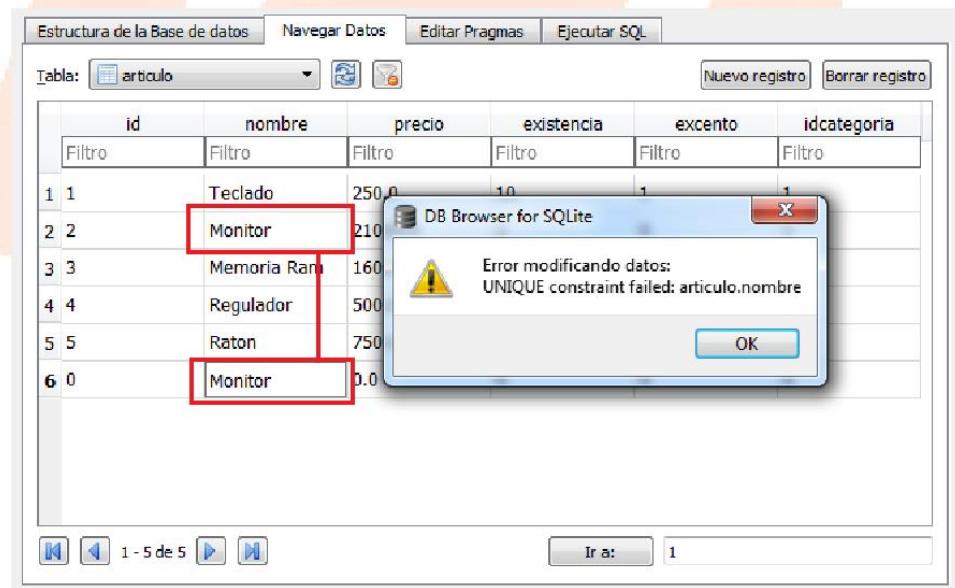


	id	nombre	precio	existencia	exento	idcategoria
	1	Teclado	250.0	10	1	1
	2	Monitor	21000.0	2	0	1
	3	Memoria Ram	16000.0	0	1	1
	4	Regulador	500.0	3	1	1
	5	Raton	750.0	7	1	1

En SQLite, para indicar que un campo es clave única, se debe acceder a la ventana de modificación de una tabla y seleccionar la columna identificada con la letra "U" (de único) del campo correspondiente:



Al especificar que un campo es una clave única, el manejador de base de datos se asegura que el valor de ese campo no se repita entre los registros, creando así una validación de los datos y logrando evitar la redundancia. Al intentar agregar un valor que viola la regla de unicidad se muestra un error como el siguiente:



La determinación de cuales campos son claves únicas o candidatas depende de cada caso. En algunas tablas un campo será único pero en otras tablas no. En el ejemplo anterior, en algunos casos el campo "nombre" podría ser un campo único si el nombre de los clientes no se puede repetir, no es el caso de los nombres de empresas, pero si en esa misma tabla se almacenará nombres de personas, entonces no podría ser clave única.

6.3.- Clave Candidata Compuesta

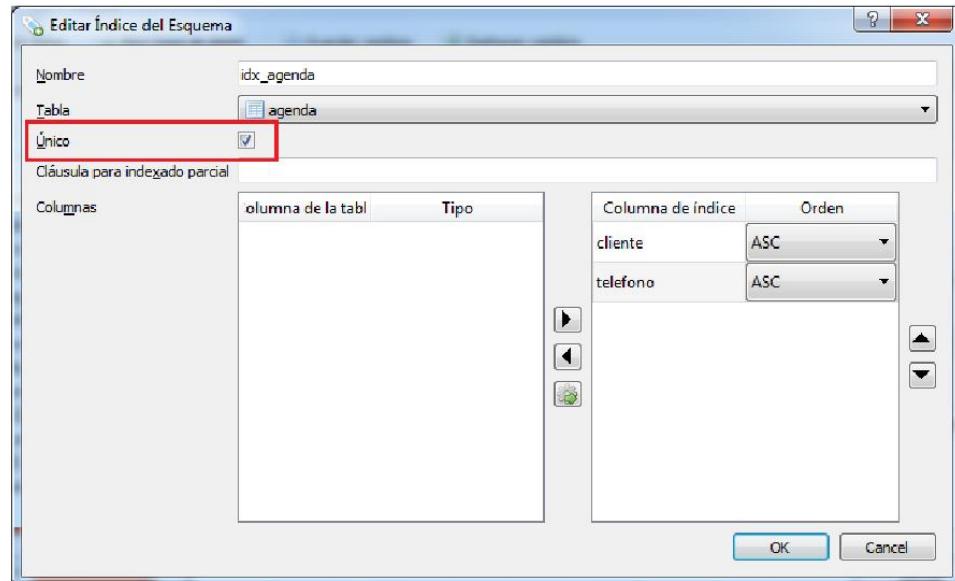
En ciertas circunstancias cuando se definen los campos de una tabla, ningún campo por si sólo es clave candidata o única, dado que los valores de ese campo se pueden repetir entre los registros. Sin embargo, la combinación de valores de 2 campos o más no se puede repetir. En ese caso, la clave única es la combinación de esos campos. A esto se le llama "clave única compuesta". Un ejemplo de una tabla con clave candidata compuesta es el siguiente:

cliente		telefono
Filtro	Filtro	
1	jose	04125806320
2	jesus	04245612310
3	maria	04268511230
4	jose	04248512020
5	luis	04125806320

En el ejemplo anterior de la agenda telefónica, el campo "nombre" del contacto se puede repetir, porque una persona puede tener varios números de teléfono. El campo "telefono" se puede repetir, asumiendo que un número de teléfono lo

pueden tener varias personas (en algunos modelos de negocio no es así), pero el nombre del contacto combinado con el número de teléfono no se debe repetir.

En SQLite para establecer una clave única compuesta se debe crear un índice agregando los campos cuya combinación no se puede repetir y luego marcar el checkbox "único", como se muestra en la imagen:

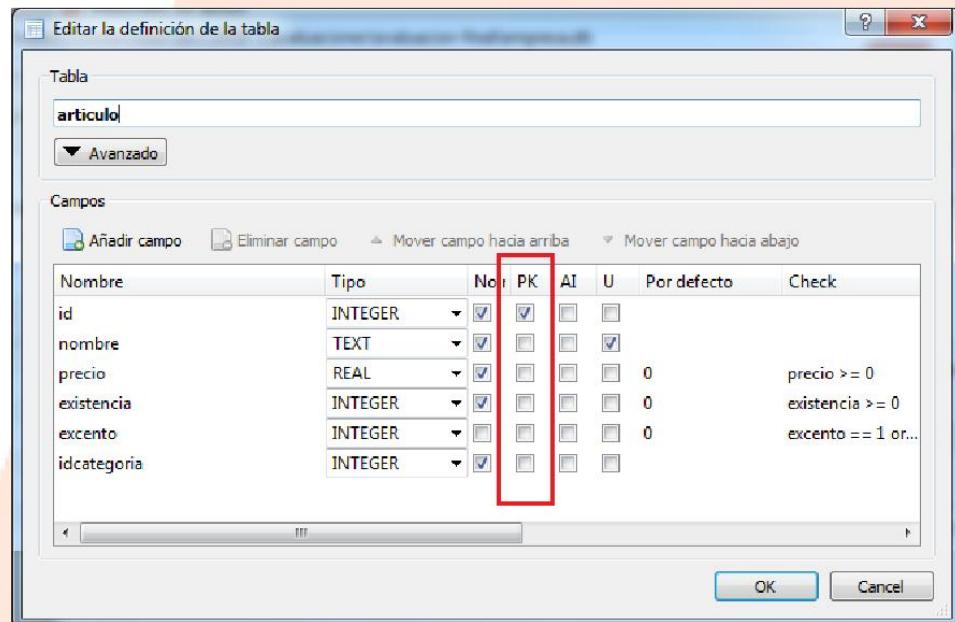


No se debe confundir una clave candidata compuesta con el hecho de tener varios campos que sean clave candidata por sí solos.

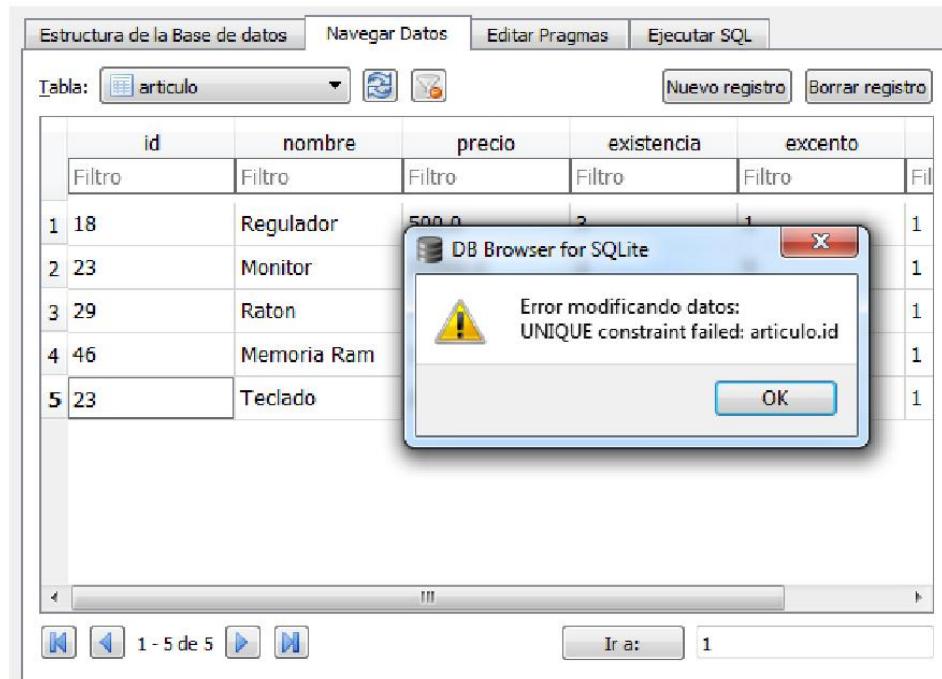
Capítulo 7. CLAVE PRIMARIA

7.1.- Clave Primaria

La clave primaria o clave principal (Primary Key en inglés) es una de las claves candidatas o únicas de la tabla que se considera el dato más representativo de cada registro para identificarlo individualmente del resto de los registros. Cuando hay una sola clave candidata, ese campo debería ser la clave primaria. Si hay varias claves candidatas, se debe decidir cuál de éstas claves es la más relevante de todas. En SQLite, al crear la tabla o modificarla, el campo que sea la clave primaria se le debe marcar el checkbox "PK" en la fila correspondiente:

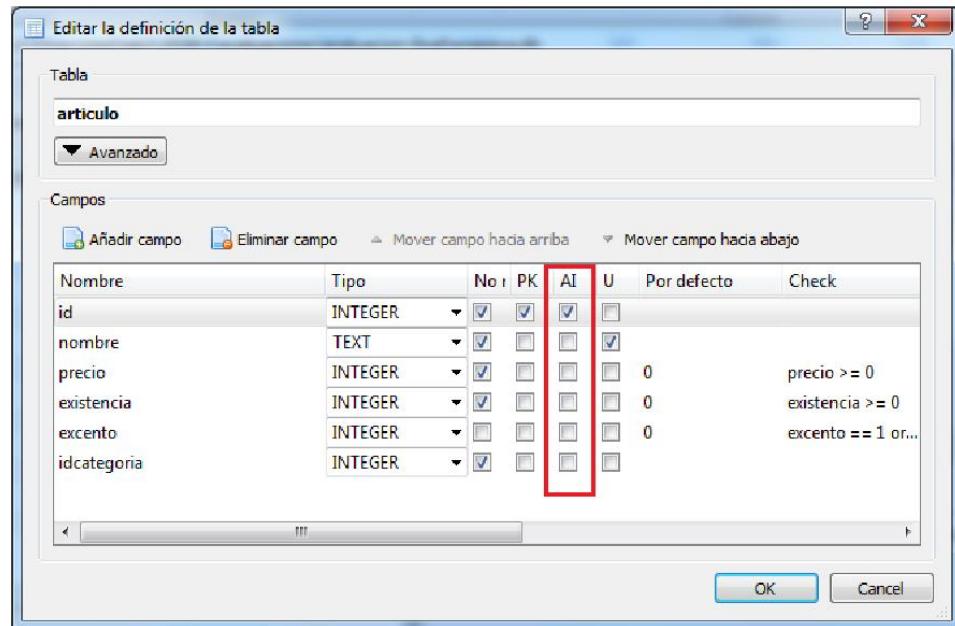


Cuando un campo se establece como clave primaria automáticamente se crea una restricción de valor único, sin necesidad de marcar el campo como único. Las claves primarias se utilizarán internamente como índices, para optimizar las búsquedas de registros y también más adelante para relacionar las tablas unas con otras. En la siguiente imagen se muestra como se verifica que el valor de una clave primaria no se repita:



7.2.- Campos Auto numéricos

Cuando una tabla no tiene una clave única, es una práctica muy común agregar un campo adicional numérico cuyo valor se genera automáticamente al agregar un nuevo registro en la tabla. A estos campos se les llama campos auto numéricos o auto incrementales (el nombre varía de un manejador de base de datos a otro). Los campos auto numéricos en la mayoría de bases de datos deben ser establecidos obligatoriamente como clave primaria. En SQLite, para hacer que un campo sea auto numérico se le marca el checkbox "AI":



No puede haber 2 campos auto numéricos en una misma tabla. Un campo auto numérico por su misma naturaleza auto generada, no se le solicita al usuario en las interfaces o formularios; generalmente ni siquiera es mostrado en las porque es de uso interno. En SQLite, un campo debe ser de tipo "Integer" para que sea auto numérico. Al crear un nuevo registro, el campo marcado como auto numérico será inicializado con el nuevo valor, tomando en cuenta el valor más alto de los registros que ya existan en la tabla:

	id	nombre	precio	existencia	excento	
	Filtro	Filtro	Filtro	Filtro	Filtro	Fil
1	18	Regulador	500	3	1	1
2	23	Monitor	21000	2	0	1
3	29	Raton	750	7	1	1
4	46	Memoria Ram	16000	0	1	1
5	150	Teclado	250	10	1	1
6	151		0	0	0	0

Capítulo 8. CLAVE FORÁNEA

8.1.- Clave Foránea (fk)

Una clave foránea (Foreign Key o simplemente FK) es un campo que está presente en una tabla sólo con el fin de "asociar" la tabla está con otra tabla donde ese mismo campo es clave primaria. Los diseñadores de bases de datos usan muy frecuentemente las claves foráneas debido a que esta estrategia es lo que permite cumplir con las reglas de diseño o formas normales (normalización). Las claves foráneas son fundamentales para las bases de datos relacionales, sin éstas, no habría ninguna relación entre las tablas.

Por ejemplo:

- una tabla con el nombre "cliente" cuya clave primaria es el campo "rif" (PK).
- una tabla con el nombre "factura" cuya clave primaria es el campo "numero" (PK).
- la tabla "factura" para relacionarse con la tabla "cliente" debe tener entre uno de sus campos el campo "rif" del cliente, que será la clave foránea en la tabla "factura".

El nombre de una clave foránea es irrelevante, pero generalmente se utiliza alguna nomenclatura que permita fácilmente saber de cuál tabla procede. En cuanto al tipo de dato si es indispensable que sean iguales.

En la siguiente imagen se muestran las tablas del ejemplo descrito anteriormente, resaltando la clave foránea en la tabla "factura", que es clave primaria en la tabla "cliente":

Nombre	Tipo	E
articuloxfactura	C	
categoria	C	
cliente	C	
rif	TEXT	'ri'
nombre	TEXT	'n'
direccion	TEXT	'd'
correo	TEXT	'c'
factura	C	
numero	INTEGER	'n'
fecha	TEXT	'fi'
rif	TEXT	'ri'
monto	REAL	'n'
sqlite_sequence	C	
usuario	C	
Índices (2)		

Es importante destacar que el campo "rif" en la tabla "factura" es una clave foránea conceptualmente hablando, pero para el manejador de base de datos es un campo como cualquier otro hasta que no se especifique expresamente que ese campo es una clave foránea. Este proceso que varía de un manejador de bases de datos a otro.

8.2.- Fk y Campos Auto numéricos

Cuando la clave primaria es un campo auto numérico, en las tablas donde éste campo será clave foránea no será auto numérico, será un simple campo de tipo "integer". Por ejemplo:

Editar la definición de la tabla					
Título: pais					
Campos					
Añadir campo	Eliminar campo	Mover campo hacia arriba	Mover campo hacia abajo		
Nombre	Tipo	No	PK	AI	U
id	INTEGER		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
nombre	TEXT		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Editar la definición de la tabla					
Título: cliente					
Campos					
Añadir campo	Eliminar campo	Mover campo hacia arriba	Mover campo hacia abajo		
Nombre	Tipo	No	PK	AI	U
rif	TEXT		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
nombre	TEXT		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
direccion	TEXT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
correo	TEXT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
cpais	INTEGER		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

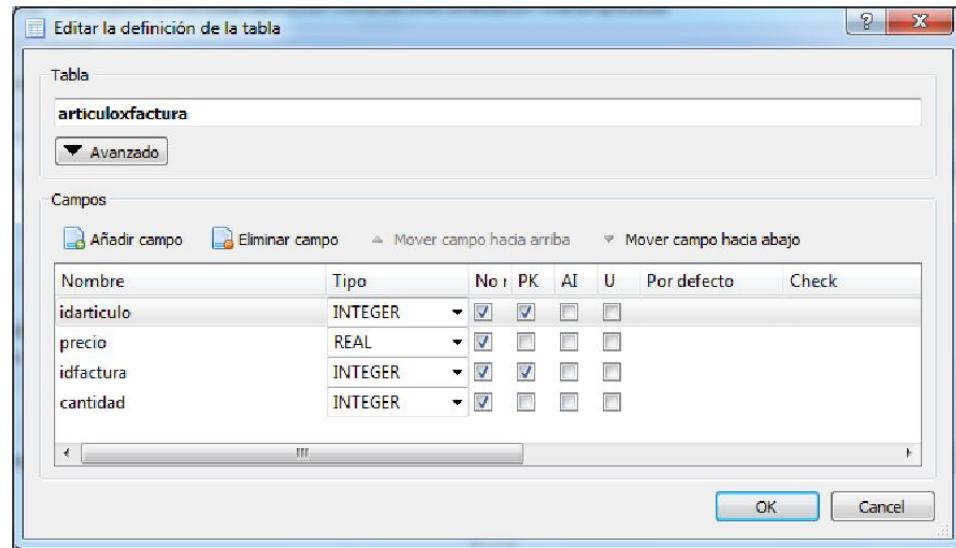
En el ejemplo anterior, el campo "id" de la tabla "país" es auto numérico, por consecuencia, cada vez que se registre un nuevo país se generará un nuevo id. Pero en la tabla "cliente" el campo "idpais" que es la clave foránea de "país", no es auto numérico, porque en ese caso, cada vez que se registra un nuevo cliente también se generará un nuevo id de país.

8.3.- Clave Primaria Compuesta

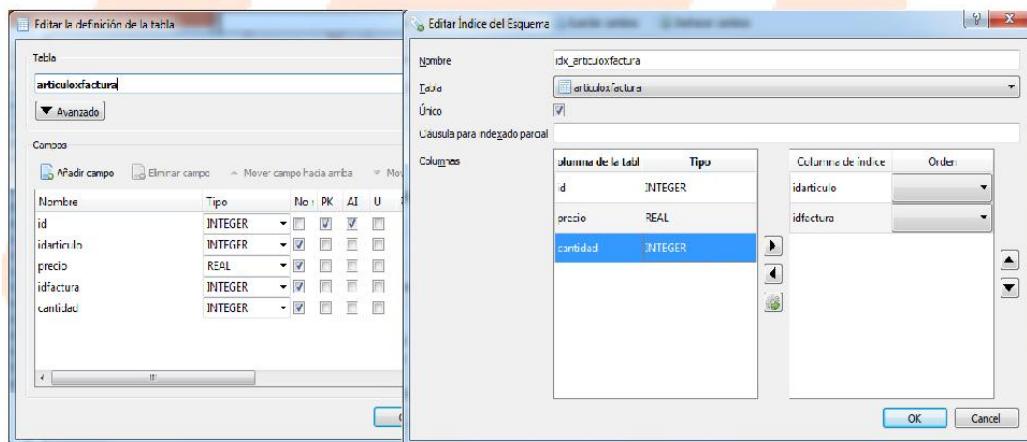
Una tabla puede tener varias claves foráneas en su lista de campos. Cuando esto ocurre es muy común establecer una clave primaria compuesta, es decir, una clave primaria donde hay 2 o más campos involucrados. Por ejemplo, en la siguiente imagen la tabla "articuloxfactura" tiene el campo "idarticulo" que es clave foránea de la tabla "articulo" y el campo "idfactura" que es clave foránea de la tabla "factura":

Estructura de la Base de datos		Navegar Datos	Editar Pragmas	Ejecutar
	Nombre	Tipo		
+	articulo			
+	id	INTEGER	id	C
+	nombre	TEXT	n	C
+	precio	INTEGER	p	C
+	existencia	INTEGER	e	C
+	exento	INTEGER	e	C
+	idcategoria	INTEGER	ic	C
+	articuloxfactura			
+	idarticulo	INTEGER	id	FK
+	precio	REAL	p	C
+	idfactura	INTEGER	id	FK
+	cantidad	INTEGER	c	C
+	categoria	CHAR	C	C
+	cliente	CHAR	C	C
+	factura	CHAR	C	C
+	id	INTEGER	id	
+	numero	INTEGER	n	C
+	fecha	TEXT	f	C
+	rif	TEXT	rif	C
+	monto	REAL	m	C

Según las reglas de negocio, es posible establecer que la combinación de valores de ambas claves foráneas no se repita, por lo tanto se puede definir como clave primaria la combinación de ambos campos. En la siguiente imagen se muestra el ejemplo:



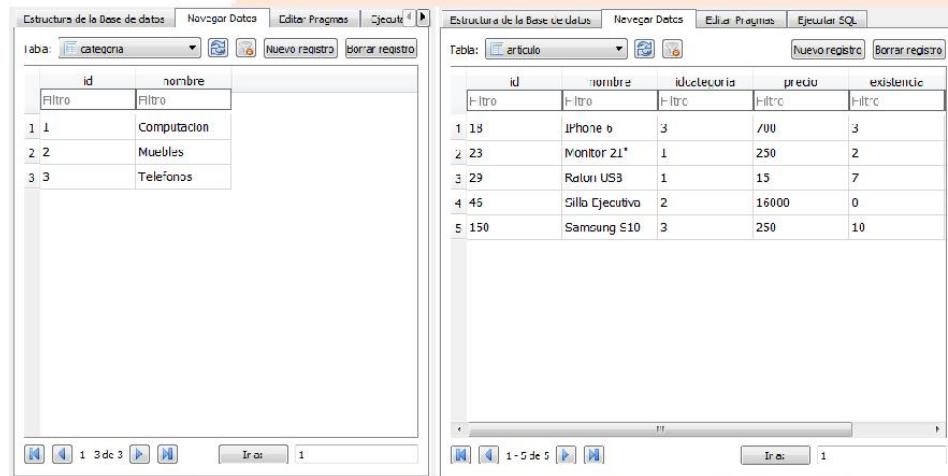
También es una práctica muy común agregar un campo auto numérico como clave primaria y crear un índice de clave compuesta única combinando los campos para lograr la restricción que cumpla con la regla de negocio, pero en ese caso no se le llamaría clave primaria compuesta. Por ejemplo:



Capítulo 9. DATOS DE CLAVES FORÁNEAS

9.1.- Datos Clasificadores

Al diseñar bases de datos se deben seguir ciertas reglas para hacer las bases de datos más eficientes. Por esta razón principal por la que se crean tantas tablas en una base de datos y existen tantas claves foráneas. Un clasificador, como su nombre lo indica, permite clasificar los registros bajo algún criterio. Por ejemplo:



id	nombre
1 1	Computación
2 2	Muebles
3 3	Teléfonos

id	nombre	idcategoria	precio	existencia
1 13	iPhone 6	3	700	3
2 23	Monitor 21"	1	250	2
3 29	Reloj US\$3	1	15	7
4 45	Silla Ejecutiva	2	16000	0
5 150	Samsung S10	3	250	10

En el ejemplo anterior, los artículos que se venden en una tienda se clasifican en una categoría, por tal razón existe una tabla "categoría" que almacena las categorías de los artículos. La tabla que almacena los artículos tiene los campos con los datos básicos y tiene también una clave foránea que hace referencia a la categoría de ese artículo. Según los datos de la tabla "articulo", los artículos con id 18 y 150 son de la categoría "Teléfonos", los artículos con id 23 y 29 son de la categoría "Computación" y el artículo con id 46 es de la categoría "Muebles".

Esta estrategia se utiliza para varios fines:

- evitar redundancia.
- reducir inconsistencia.
- filtrar registros.
- generar estadísticas.

9.2.- Datos Para Registros Múltiples

Al momento de necesitar guardar varios datos asociados a un registro no se guardan separados por comas en un mismo campo, se utiliza otra tabla. Por ejemplo, si se necesitan almacenar los números de teléfonos de los clientes, debe existir una tabla con los datos de los clientes y otra tabla para los números de teléfonos, ésta última con una clave foránea de la tabla cliente. La siguiente imagen las tablas:

Nombre	Tipo
Tablas (9)	
agenda	
rifcliente	TEXT
telefono	TEXT
articulo	C
articuloxfactura	C
categoria	C
cliente	C
rif	TEXT
nombre	TEXT
direccion	TEXT
correo	TEXT
idpais	INTEGER
factura	C
pais	C
sqlite_sequence	C
usuario	C
Índices (2)	
idx_agenda	C
idx_articulo	C
Vistas (0)	

Este caso no es técnicamente diferente al de los clasificadores, pero tiene una connotación diferente a nivel de diseño y de los datos. La siguiente imagen muestra un ejemplo de posibles valores que pueden tener las tablas:

Estructura de la Base de datos					Navegar Datos	
Tabla: cliente					Nuevo registro	Borrar registro
rif	nombre	direccion	correo	id	Filtro	Filtro
1 4045640	Blanca Dellan	Av Pedro Leon	jose@caid	NULL		
2 3488027	Luis Rojas	CC Las trinitar...		NULL		
3 14978130	Jesus Perez	CC Sotavento	NULL	NULL		

Estructura de la Base de datos		Navegar Datos	
Tabla: agenda		Nuevo registro	Borrar registro
rifcliente	telefono	Filtro	Filtro
1 14978130	04125806320		
2 4045640	04245612310		
3 4045640	04268511230		
4 3488027	04248512020		
5 14978130	04125806322		

En el ejemplo anterior el cliente de nombre Blanca Dellan tiene 2 números de teléfono asociado (04245612310 y 04268511230), el cliente Luis Rojas tiene asociado sólo un número de teléfono (04248512020) y el cliente de nombre Jesus Perez tiene asociado 2 números de teléfono (04125806320 y 04125806322).

9.3.- Datos Con Múltiples Claves Foráneas

Otra situación muy común es la que se presenta al momento de guardar varios datos asociados a un registro, que a su vez están asociados con registros de otra(s) tabla(s). Por ejemplo, una factura tiene asociados varios productos. El diseño más común para dar solución a esta situación es una tabla donde se almacenan los datos básicos de la factura y otra donde se almacenan los artículos facturados, pero los datos de los productos están en una tabla aparte.

The screenshot shows three database tables in MySQL Workbench:

- producto** table:

id	codigo	nombre
1	451424	S5
2	541214	Tachipirin
3	211414	Plasma 50"
4	25541	Iphone 10

- productoxfactura** table:

id	idproducto	numerofactura	cantidad
1	5	1	5
2	6	1	1
3	7	2	3
4	8	1	2

- factura** table:

numero	fecha	monto
1	10/10/2018	15623.0
2	18/12/2018	16554

En el ejemplo anterior, la factura número 1 tiene asociados 3 productos o artículos, que son:

- 2 unidades de S5 (id =1),
- 5 unidades Tachipirin (id = 2)
- 1 unidades Plasma 50" (id = 3)

La factura número 2 tiene asociado un sólo producto:

- 3 unidades Tachipirin (id = 2)

Capítulo 10. INTEGRIDAD REFERENCIAL

10.1.- Integridad Referencial

La integridad referencial es una propiedad de las bases de datos, que obliga a que la clave foránea de una tabla haga referencia siempre a una fila válida de la tabla de la tabla donde es clave primaria. La integridad referencial garantiza que la relación entre dos tablas permanezca sincronizada durante las operaciones CRUD y por lo tanto que los datos que están almacenados en éstas tengan la mayor coherencia y utilidad posible.

Por ejemplo, si una tabla almacena las categorías de los artículos que se venden en una tienda y otra tabla almacena los datos de los artículos, donde uno de los campos de la tabla artículos es una clave foránea que hace referencia a la categoría del artículo, la integridad referencial obliga que al crear un nuevo registro en la tabla artículo se haga referencia a una categoría existente en la tabla categoría, de lo contrario, se obtendrá un error y no se permite la operación.

También, cuando se intente eliminar un registro en la tabla categoría, la integridad referencial exige que no exista ningún artículo asociado a ésta, es decir, se comprueba que no existe ninguna referencia en la tabla artículo a la categoría que se intenta borrar, de lo contrario, no se permite la operación o se hace una eliminación en cascada.

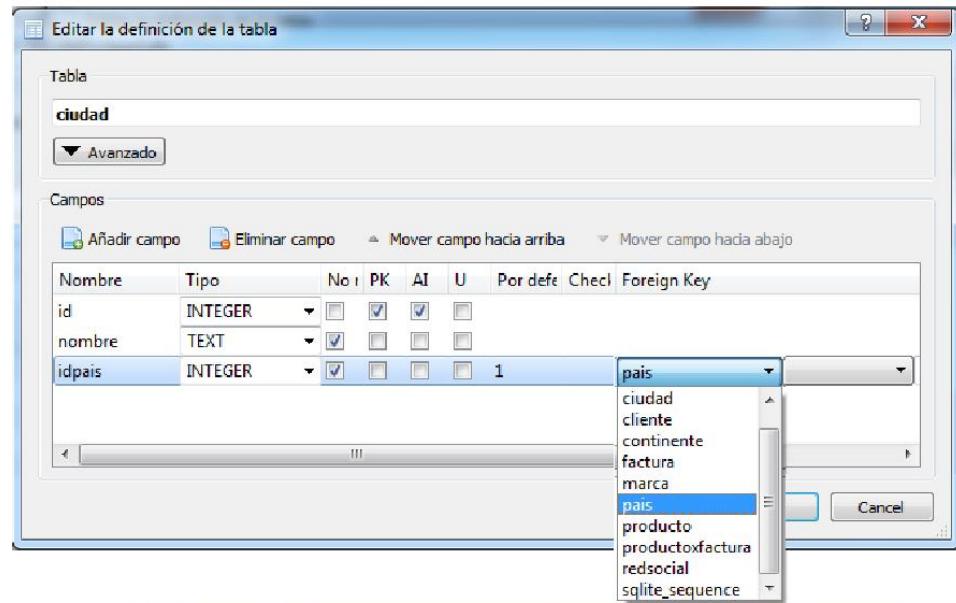
En el siguiente ejemplo hay 2 tablas: categoría y producto. En la tabla producto hay una clave foránea que hace referencia a la tabla categorías. Al verificar los datos se puede notar que no hay integridad referencial debido a que en algunos registros de la tabla producto se hace referencia a valores de id de categoría que no existen en la tabla categoría:

	id	nombre		
	Filtro	Filtro		
1	1	Computacion		
2	2	Línea Blanca		
3	3	Frutas		
4	4	Viveres		
5	5	Hortalizas		
6	6	Electricidad		
7	7	Medicinas		
8	8	Juguetes		
9	9	Telefono		
10	10	Televisores		

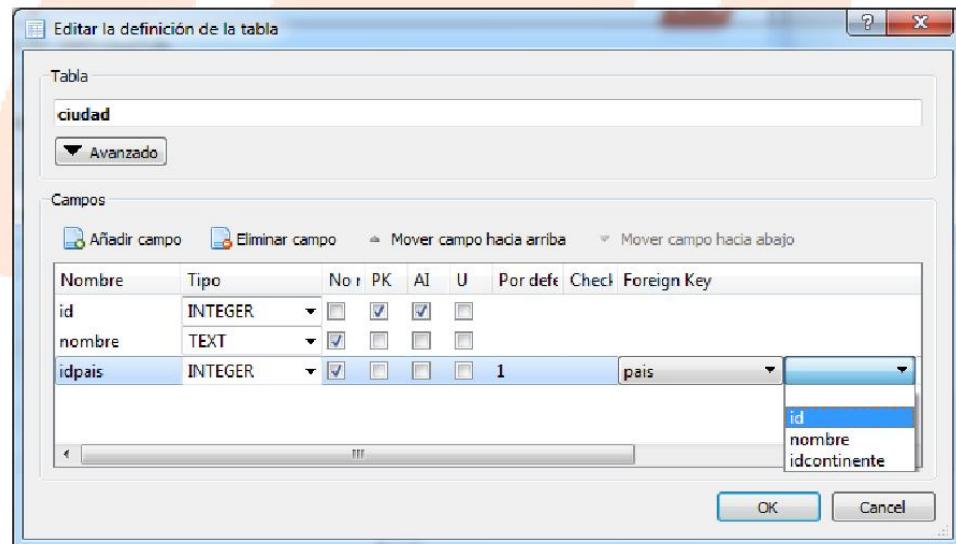
	nombre	existencia	idcategoria	idmarca
	Filtro	Filtro	Filtro	Filtro
1	S5	0	9	1
2	Tachipirin	0	20	8
3	Plasma 50"	0	12	1
4	Iphone 10	0	9	3

10.2.- Establecer Claves Foráneas

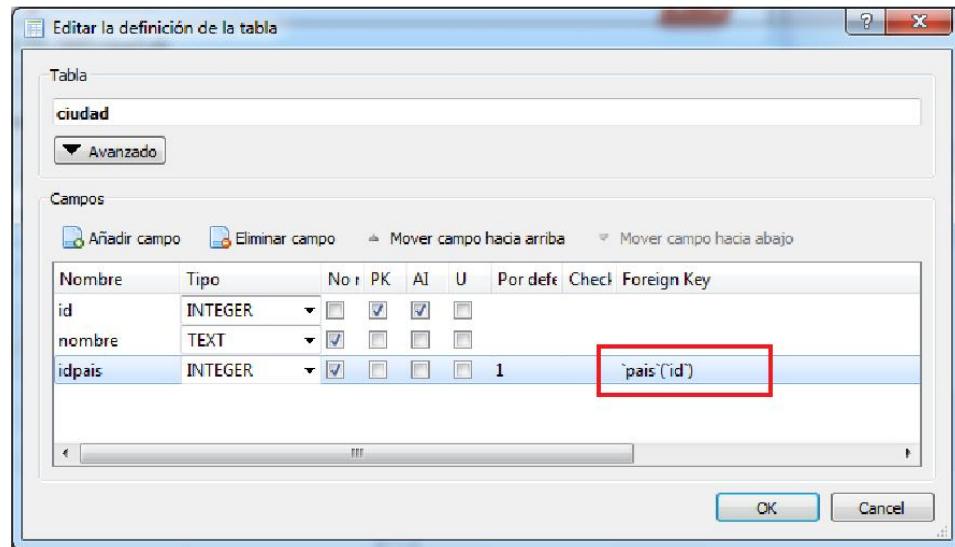
Como se explicó anteriormente, un campo no es una clave foránea técnicamente hablando hasta que se especifique en el manejador de base de datos. DB Browser for SQLite tiene en los atributos de un campo una columna adicional con el nombre "Foreign Key", donde se especifica la tabla de la cual proviene la clave foránea y cuál es la clave primaria de esa tabla. La siguiente imagen muestra un ejemplo:



Al seleccionar la tabla de origen, se muestran los campos que contiene la tabla seleccionada. En el ejemplo anterior, la tabla "ciudad" tiene un campo con el nombre "idpais", que es una clave foránea que hace referencia a la tabla "pais", cuya clave primaria es el campo "id".

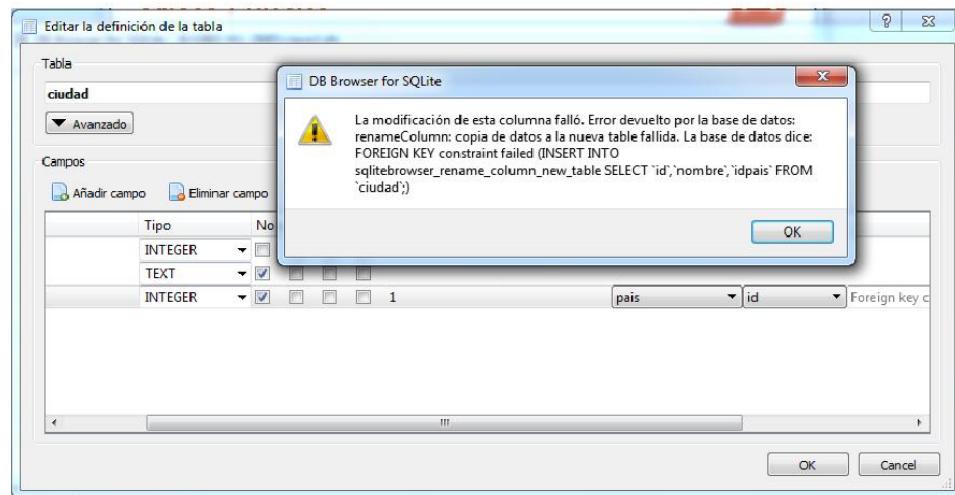


Debe asegurarse que al seleccionar la tabla y el campo quede establecido en la columna "Foreign Key" un valor que contenga el nombre de la tabla y el nombre del campo, en el formato "tabla(clave)", como se muestra en la imagen:



10.3.- Validación de Los Datos

Al establecer la clave foránea, el manejador de base de datos empezará a hacer validaciones de integridad referencial. Lo primero que verificará es si los registros que ya existen en la tabla al momento de establecer la clave foránea cumplen con la restricción, en caso contrario se generará un error y no se podrá guardar los cambios. Por esta razón se recomienda realizar esta tarea antes de agregar los primeros registros en la tabla. El error que se muestra es el siguiente:



Si los datos cumplen con las restricciones de clave foránea, se guardan los cambios y se empezará a validar de ahí en adelante cada vez que se intente agregar un registro nuevo a la tabla. En el siguiente ejemplo se intenta asignar al campo "idpais" un valor que no existe en el campo "id" de ningún registro de la tabla "pais":

	id	nombre	idcontinente
1	1	Venezuela	2
2	2	Colombia	2
3	3	Sudáfrica	1
4	4	España	4
5	5	China	3
6	6	India	3
7	7	Portugal	4
8	8	Francia	4

	id	nombre	idpais
1	1	Barquisimeto	1
2	2	Caracas	20