AGO
31

SPRINT 3 - MEETING 47

Base de datos relacionales I

Grabación

Presentación

Introducción a bases de datos relacionales

> *"La ciencia de datos no es necesariamente una sola disciplina, conjunto de habilidades o metodología. Es por eso que siempre se dice que la ciencia de datos es una 'rama interdisciplinaria' de la ciencia que combina matemáticas, análisis de comportamiento humano y estudios de flujo de trabajo, uso flexible de sistemas lógicos y un empleo básico de algoritmos." — Adrian Bridgwater.* *Periodista de tecnología dedicado al desarrollo de aplicaciones de software 'beat'.*

designed by freepik

En la bitácora anterior aprendiste sobre bases de datos. Conociste diferentes métodos y —con foco en las bases de datos no relacionales— diste los primeros pasos con Mongo DB. En esta bitácora te

relacionales (BDR). Conocer ambos métodos es importante para saber bien cuál elegir según nuestras necesidades al momento de desarrollar una aplicación.

Según la página oficial de Oracle, en los orígenes, cuando se empezaban a usar bases de datos, cada aplicación los almacenaba en su propia estructura única y sucedía:

"Cuando los desarrolladores querían crear aplicaciones para usar esos datos, tenían que saber mucho sobre la estructura de datos particular para encontrar los datos que necesitaban. Estas estructuras de datos eran ineficientes, difíciles de mantener y difíciles de optimizar para ofrecer un buen rendimiento de la aplicación. El modelo de base de datos relacional se diseñó para resolver el problema de varias estructuras de datos arbitrarias. El modelo relacional proporcionó una forma estándar de representar y consultar datos que cualquier aplicación podría utilizar. Desde el principio, los desarrolladores reconocieron que la principal fortaleza del modelo de base de datos relacional estaba en el uso de tablas, que eran una forma intuitiva, eficiente y flexible de almacenar y acceder a información estructurada."

Vamos a ampliar la definición. Según Wikipedia, las bases de datos se utilizan desde los años '70, cuando Edgar Frank Codd (de los laboratorios IBM en San José, California) postuló sus bases. Éstas no tardaron en consolidarse como un nuevo paradigma en los modelos de base de datos y son una de las formas más utilizadas por los/as desarrolladores/as.

Se pueden utilizar, por ejemplo, para hacer seguimiento de inventarios, procesar transacciones de comercio electrónico o administrar grandes cantidades de información. Se puede aplicar una base de datos relacional para cualquier necesidad de información en la que los puntos de datos se relacionan entre sí y se deban administrar de una manera segura, consistente y basada en reglas.

Cómo entender las bases de datos relacionales

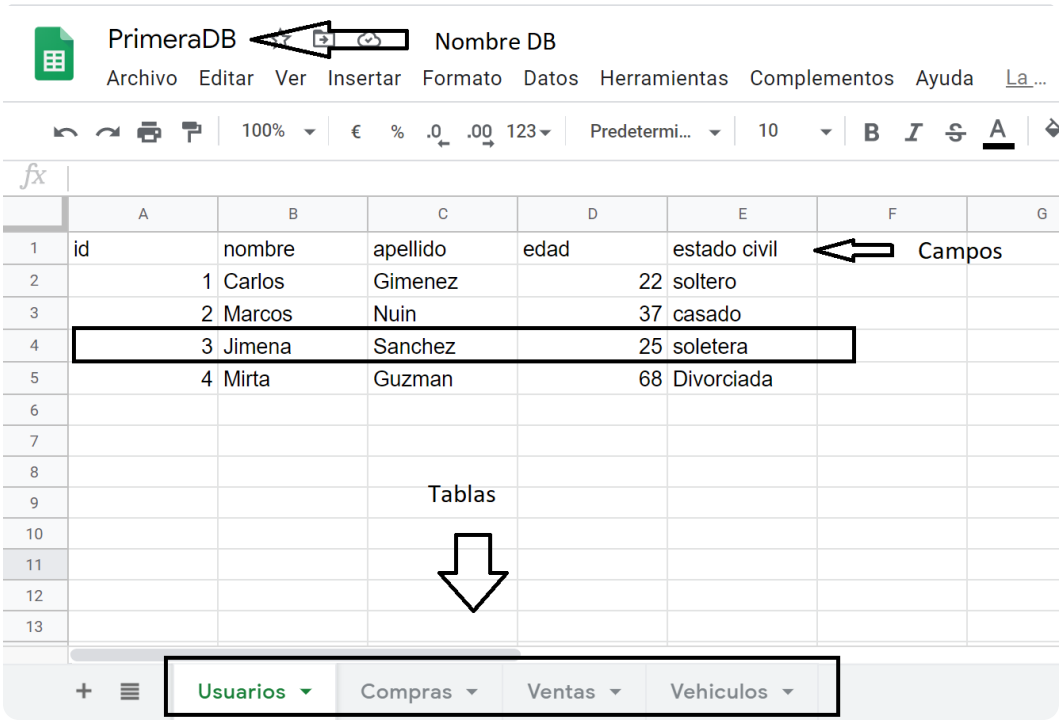
Para empezar a hablar sobre bases de datos relacionales, es clave conocer a qué nos referimos con los siguientes **conceptos:**

- **Base de datos.** Es el nombre que engloba toda tu información.

- **Campos.** Es el nombre que identifica cada uno de los datos en tus entidades.
- **Registros.** Es un conjunto de datos almacenados.
- **Consultas SQL.** Es el lenguaje a través del cual podemos interactuar con la base de datos.

Si imaginas tu base de datos como si fuera un archivo Excel, las relaciones serían las siguientes:

- **Base de datos:** el nombre del archivo de excel.
- **Tabla:** el nombre del libro.
- **Campos:** cada una de las columnas.
- **Registros:** cada una de las filas completas de información.



Cómo crear una base de datos relacional

1. **Planifica el tipo de información que se necesita almacenar,** teniendo en cuenta la información disponible y la que necesitamos.

2. **Crea una tabla por cada entidad que tengas en el sistema.** Entendemos por entidad a un objeto que vamos a modelar, del cual necesitamos sus atributos. Para cada campo es necesario determinar los tipos de datos que va a contener, aglomerados en 3 grandes grupos: numéricos, fecha y cadena.

a. **Tipos numéricos**

Tipo de Campo	Rango de valores	Tamaño
TINYINT	-128 a 127	1 byte
SMALLINT	-32768 a 32767	2 bytes
MEDIUMINT	-8.388.608 a 8.388.607	3 bytes
INT	-2147483648 a 2147483647	4 bytes
BIGINT	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	8 bytes
FLOAT	1.175494351E-38 a 3.402823466E+38	4 bytes
DOUBLE	2.2250738585072014E-308 a 1.7976931348623157E+308	8 bytes

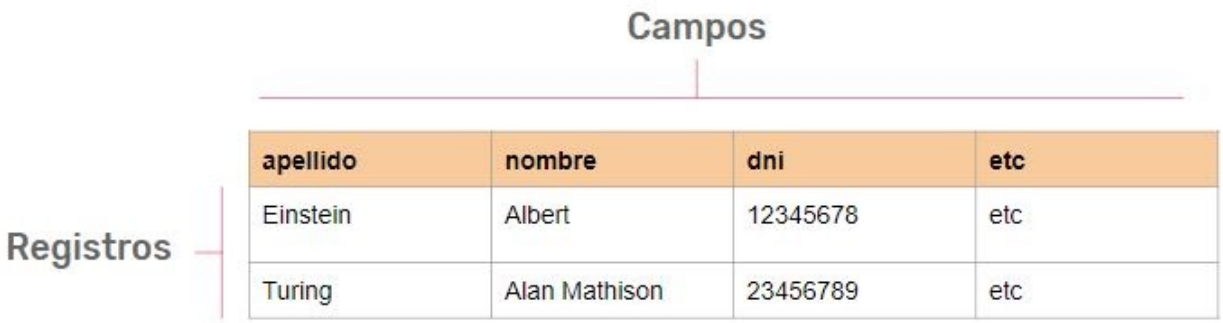
b. **Tipos de Fecha**

Tipo de Campo	Rango	Tamaño
DATE	1 de enero del 1001 al 31 de diciembre de 9999	3 bytes
DATETIME	1 de enero del 1001 al 31 de diciembre del 9999	8 bytes
TIMESTAMP	1 de enero de 1970 al año 2037	4 bytes
TIME	-838 horas, 59 minutos y 59 segundos a 838, 59 minutos y 59 segundos	3 bytes
YEAR	1901 al año 2155	1 byte

c. **Tipos de Cadena**

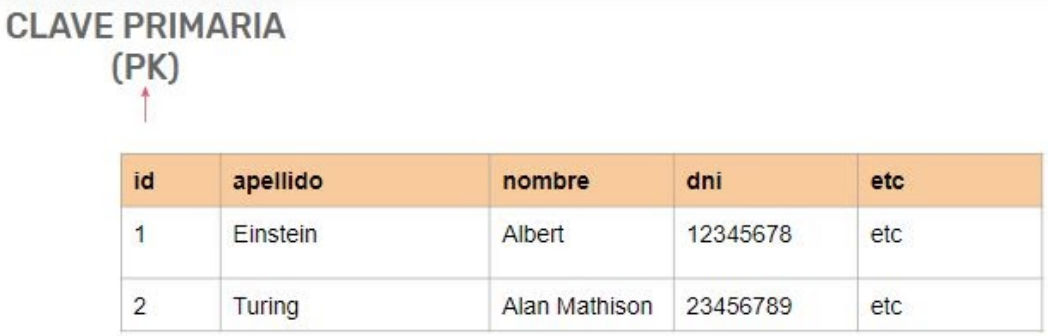
Tipo de campo	Tamaño de Almacenamiento
CHAR(n)	n bytes
VARCHAR(n)	n +1 bytes
TINYBLOB, TINYTEXT	Longitud+1 bytes
BLOB, TEXT	Longitud +2 bytes
MEDIUMBLOB, MEDIUMTEXT	Longitud +3 bytes
LONGBLOB, LONGTEXT	Longitud +4 bytes
ENUM('value1','value2',...)	1 ó dos bytes dependiendo del número de valores
SET('value1','value2',...)	1, 2, 3, 4 ó 8 bytes, dependiendo del número de valores

Por ejemplo, imagina que cuentas con una base de datos de “científicos”. Estos podrían ser los componentes en tu tabla:



En este caso, las columnas contienen las propiedades de la entidad que estamos describiendo y en las filas el contenido de cada uno de los registros. Tendríamos el apellido, nombre, DNI y otros datos que quisiéramos agregar.

Otro concepto importante es el de **clave primaria:** es una columna especial que permite identificar cada uno de los registros de forma única. Para evitar que haya datos repetidos, agregamos la columna adicional ID al inicio de cada tabla y le asignamos la propiedad auto_increment . Así, se le indica al motor de la base de datos que tiene que aumentar ese valor de forma consecutiva cada vez que se agregue un nuevo registro. Esto nos garantiza que nunca se va a repetir en dos registros por más que el resto de los datos sean iguales:



id	apellido	nombre	dni	etc
1	Einstein	Albert	12345678	etc
2	Turing	Alan Mathison	23456789	etc

Cuando definimos una clave primaria, ésta se convierte en un índice que funciona exactamente igual que el índice de un libro. Es lo que va a permitir al motor de la base de datos que, cuando intentemos hacer una consulta por ID, sea mucho más rápido el proceso de acceder y encontrar los registros. Se puede definir como índice cualquiera de las otras columnas de nuestra tabla, pero esto puede tener consecuencias tanto positivas como negativas. De todas maneras, sigue siendo mucho más rápido que si utilizáramos las escrituras, por ejemplo.

Aprender un lenguaje de consulta estructurado

De las formas que existen de administrar información relacional, nosotros nos vamos a centrar en aprender **SQL (Structured Query Language).** Es un lenguaje estándar creado para guardar, manipular y consultar bases de datos relacionales que se basa en el álgebra relacional. Esto quiere decir que proporciona un lenguaje matemático internamente consistente, que facilita la mejora del rendimiento de todas las consultas de la base de datos. Según el artículo de Marlon Marzon podemos encontrar las siguientes ventajas:

- **Madurez:** dado que tiene ya muchos años y aceptación por la comunidad de developers, existe una gran variedad y cantidad de información para poder realizar cualquier tipo de desarrollo o extracción de información. Esto ayuda increíblemente en la mejora de tiempos de entrega de cualquier proyecto de software.
- **Atomicidad:** es la propiedad que garantiza que cualquier operación realizada en la base de datos no se vea afectada si llega a ocurrir algún problema en la mitad del proceso, ni que quede a medias.

- **Estándares bien definidos:** por ejemplo, la creación de tablas, insertar, eliminar y actualizar información, consultas, se escriben bajo la misma sintaxis, basada en el estándar de SQL.
- **Sencillez en la escritura:** ya que se asemeja mucho al lenguaje humano, la comprensión de las operaciones que se programen pueden ser interpretadas o escritas por personas que no tengan grandes conocimientos de informática.

Mediante este lenguaje vamos a poder hacer consultas a nuestras tablas en nuestra base de datos.

Cómo administrar la base de datos

phpMyAdmin es un software que te permitirá administrar bases de datos Maria DB. Desde allí podrás ejecutar cualquier instrucción tipo SQL. Esto es de gran ayuda para empezar a incorporar la sintaxis y aprender el nuevo lenguaje de consultas.

Puedes utilizar un servicio gratuito que te provee no solo una base de datos, sino también un panel phpMyAdmin para su administración. Crea tu cuenta para comenzar a escribir código SQL.

En el próximo encuentro, con la ayuda de tus mentores instalarás Xampp para correr localmente el motor de base de datos junto con el PHPMyAdmin, pero si deseas, ¡puedes instalarlo ahora!

Los 7 comandos más utilizados en SQL

Nos enfocaremos en los 7 comandos más utilizados y los organizaremos en dos categorías: los que pertenecen al lenguaje de definición de datos y los que pertenecen a la manipulación de datos.

1. **Definición de datos.**

2. **CREATE TABLE.** Permite crear tablas. para eso, debes definir un nombre y los campos junto con su tipo.

```
CREATE TABLE `usuarios` (
    `nombre` varchar(100),
    `apellido` varchar(200),
    `edad` int(5)
);
```

- **ALTER TABLE.** Permite modificar la estructura de una tabla. En el siguiente ejemplo, ha cambiado el campo de nombre a nombre_completo. ¡Esta operación no modificará ningún registro!

```
ALTER TABLE usuarios CHANGE nombre nombre_completo VARCHAR(100);
```

- **DROP TABLE**. Esta instrucción elimina la tabla junto con los campos y cada uno de los datos que tenga allí almacenados.

```
DROP TABLE usuarios
```

2. **Manipulación de datos.**

3. **SELECT.** Consulta y trae información de nuestras tablas. Nos permite traer todos los registros de una tabla específica junto con todas las columnas de cada uno o utilizar el comodín * para seleccionar todos los campos

```
SELECT * FROM entidad
SELECT campo1, campo2, campo3 FROM entidad
```

- **INSERT.** Introduce nuevos registros en nuestra tabla. Debemos introducir entre paréntesis y separado por coma todos los campos donde deseo insertar un dato

luego la palabra reserva VALUES y nuevamente entre paréntesis los valores a insertar

```
INSERT INTO usuarios (nombre, apellido, edad) VALUES
(“Gustavo”, “Fernandez”, 25);
```

- **UPDATE.** Actualiza uno o varios registros en nuestra tabla.

```
UPDATE usuarios SET nombre = “Marcos”, apellido=”Sandoval”
```

- **DELETE.** Elimina registros en nuestra tabla.

```
DELETE FROM usuarios
```

- **WHERE.** La palabra reservada WHERE nos permite añadir condiciones a nuestros sentencias SQL para seleccionar registros según la condición que escribamos. Podemos utilizar WHERE combinado con el SELECT, UPDATE o DELETE.

```
SELECT * FROM usuarios WHERE
```

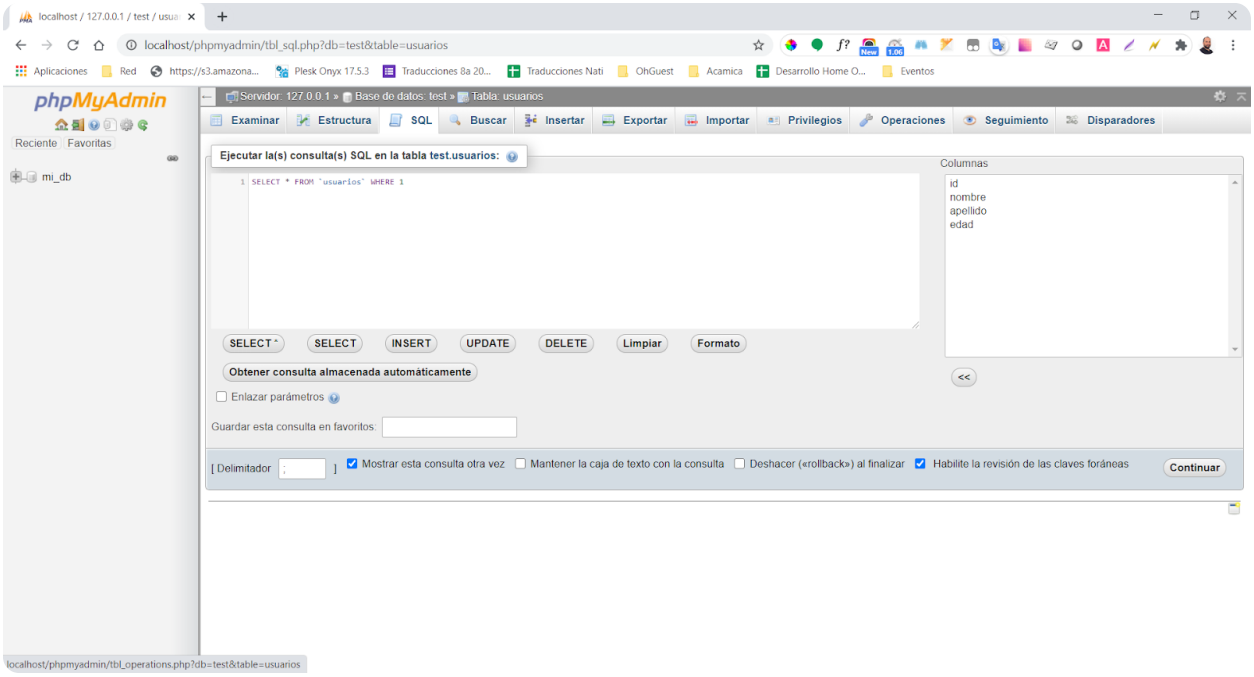
Importante

Es fundamental utilizar WHERE tanto en las sentencias DELETE cómo UPDATE, de no hacerlo estarás afectando a todos los registros de la tabla y es muy probable que no quieras hacer esto.

Ya conoces operadores para seleccionar registros. En el ejemplo anterior seleccionamos todos los

> **mayor>= mayor o igual< menor<= menor o igual= igual!= distinto**

¡A través de la consola de PHPMyAdmin podrás ejecutar todas las instrucciones SQL!



Cierre

¡Felicitaciones! Ya has dado tus primeros pasos con un nuevo lenguaje para administrar información en bases de datos relacionales. Además, empezaste a conocer SQL, que tiene su permanencia y reputación entre los/as developers. ¿Qué tal si seguimos profundizando la próxima?

¡Prepárate para el próximo encuentro!

Profundiza

Te invitamos a conocer más sobre el tema de esta bitácora.

Challenge

Te proponemos el siguiente desafío, ¿te animas?

Potencia tu Talento - Entrevistas Técnicas

Tips para construir tu perfil profesional.