Programación Web con PHP y MySQL

Unidad 1: Introducción a las bases de datos

Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 1: Introducción a las bases de datos

Objetivos

Que el alumno logre:

- Aprender los conceptos de tabla, columna, fila o registro y valor de un dato dentro de la tabla.
- Aprender a definir una llave o clave primaria y cuál es el propósito de la misma.

MySQL

CONCEPTOS BÁSICOS.

¿Qué es una base de datos?

Una base de datos es un sistema para archivar información en una computadora cuyo propósito general es mantener información y hacer que esté disponible cuando se la solicite.

Las bases de datos son un área de la computación que recibió mucha atención debido a sus múltiples aplicaciones: bibliotecas, automatización de oficinas, diccionarios automatizados y en general cualquier programa orientado a mantener y recuperar información.

A esta altura, uno puede preguntarse ¿Qué es lo interesante de usar una Base de Datos, si yo la información también la puedo guardar en archivos de texto bien organizados?

Lo interesante de usar una base de datos como sistema para almacenar información en una computadora, en vez de usar un simple archivo de texto en donde guardamos los datos que nos interesan, es que a una base de datos se le pueden pedir datos.

En otras palabras, una base de datos es un "almacén" que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente.

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, USA. Una base de datos se puede definir como un conjunto de información relacionada que se encuentra agrupada ó estructurada.

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos.

Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

Definición de base de datos

Se define entonces brevemente a una base de datos como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

Características

Entre las principales características de los sistemas de base de datos podemos mencionar:

Independencia lógica y física de los datos.

- **Independencia Física de datos:** Es la capacidad para modificar el esquema físico sin provocar que los programas de aplicación tengan que rescribirse. Las modificaciones en el nivel físico son ocasionalmente necesarias para mejorar el funcionamiento.

- **Independencia Lógica de datos:** Es la capacidad para modificar el esquema lógico sin causar que los programas de aplicación tengan que rescribirse. Las modificaciones en el nivel lógico son necesarias siempre que la estructura lógica de la base de datos se altere.

La independencia de datos lógica es más difícil de lograr que la independencia de datos física, ya que los programas de aplicación son fuertemente dependientes de la estructura lógica de los datos a los que ellos acceden.

Redundancia mínima.

Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

Acceso concurrente por parte de múltiples usuarios.

En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un Sistema de Gestión de Base de Datos (SGBD) debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

Integridad de los datos.

Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

Consultas complejas optimizadas.

La optimización de consultas permite la rápida ejecución de las mismas.

Seguridad de acceso y auditoria.

La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra asegurada frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado.

Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

Respaldo y recuperación.

Los SGBD deben proporcionar una forma eficiente de realizar copias de seguridad de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Acceso a través de lenguajes de programación estándar.

Se refiere a la posibilidad ya mencionada de acceder a los datos de una base de datos mediante lenguajes de programación ajenos al sistema de base de datos propiamente dicho.

Sistema de Gestión de Base de Datos (SGBD)

Los Sistemas de Gestión de Base de Datos (en inglés DataBase Management System) 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. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Se trata de un instrumento que nos permite ingresar, recuperar y manejar la información contenida en la base de datos. Entendemos por manejar, la posibilidad de ejecutar las siguientes operaciones, entre muchas otras:

1. Añadir nueva información a medida que ésta va ingresando.
2. Obtener la información ordenada según determinados parámetros (por orden alfabético, según el nombre del autor, según la temática de cada libro, etc.).
3. Calcular cómputos referidos a la base (cantidad total de publicaciones, promedios periódicos de ventas, promedios según las diversas categorías, etc.).
4. Imprimir la información deseada, ya sea en forma de tablas o de gráficos de diversos tipos.

Todas estas prestaciones y muchísimas más que resultaría fatigoso enumerar, comparten una característica común que constituye la más notable diferencia respecto de la base de datos tradicional. Esta diferencia consiste en que los datos se ingresan una sola vez, de una determinada forma, y pueden luego manipularse para extraer la información ordenada y seleccionada según múltiples criterios.

La utilización de bases de datos nos proporciona las siguientes ventajas:

Control sobre la redundancia de datos:

Los sistemas de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar la falta de consistencia de datos.

En los sistemas de bases de datos todos estos ficheros están integrados, por lo que no se almacenan varias copias de los mismos datos. Sin embargo, en una base de datos no se puede eliminar la redundancia completamente, ya que en ocasiones es necesaria para modelar las relaciones entre los datos.

Consistencia de datos:

Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente. Si un dato está duplicado y el sistema conoce esta redundancia, el propio sistema puede encargarse de garantizar que todas las copias se mantienen consistentes.

Compartición de datos:

En los sistemas de ficheros, los ficheros pertenecen a las personas o a los departamentos que los utilizan. Pero en los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados.

Mantenimiento de estándares:

Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

Mejora en la integridad de datos:

La integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

Mejora en la seguridad:

La seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros.

Mejora en la accesibilidad a los datos:

Muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.

Mejora en la productividad:

El SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación.

El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel.

Mejora en el mantenimiento:

En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan.

Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.

Mejora en la integridad de datos:

La integridad de la base de datos se refiere a la validez y la consistencia de los datos almacenados. Normalmente, la integridad se expresa mediante restricciones o reglas que no se pueden violar. Estas restricciones se pueden aplicar tanto a los datos, como a sus relaciones, y es el SGBD quien se debe encargar de mantenerlas.

Mejora en la seguridad:

La seguridad de la base de datos es la protección de la base de datos frente a usuarios no autorizados. Sin unas buenas medidas de seguridad, la integración de datos en los sistemas de bases de datos hace que éstos sean más vulnerables que en los sistemas de ficheros.

Mejora en la accesibilidad a los datos:

Muchos SGBD proporcionan lenguajes de consultas o generadores de informes que permiten al usuario hacer cualquier tipo de consulta sobre los datos, sin que sea necesario que un programador escriba una aplicación que realice tal tarea.

Mejora en la productividad:

El SGBD proporciona muchas de las funciones estándar que el programador necesita escribir en un sistema de ficheros. A nivel básico, el SGBD proporciona todas las rutinas de manejo de ficheros típicas de los programas de aplicación.

El hecho de disponer de estas funciones permite al programador centrarse mejor en la función específica requerida por los usuarios, sin tener que preocuparse de los detalles de implementación de bajo nivel.

Mejora en el mantenimiento:

En los sistemas de ficheros, las descripciones de los datos se encuentran inmersas en los programas de aplicación que los manejan.

Esto hace que los programas sean dependientes de los datos, de modo que un cambio en su estructura, o un cambio en el modo en que se almacena en disco, requiere cambios importantes en los programas cuyos datos se ven afectados.

Sin embargo, los SGBD separan las descripciones de los datos de las aplicaciones. Esto es lo que se conoce como independencia de datos, gracias a la cual se simplifica el mantenimiento de las aplicaciones que acceden a la base de datos.

Aumento de la concurrencia:

En algunos sistemas de ficheros, si hay varios usuarios que pueden acceder simultáneamente a un mismo fichero, es posible que el acceso interfiera entre ellos de modo que se pierda información o se pierda la integridad. La mayoría de los SGBD gestionan el acceso concurrente a la base de datos y garantizan que no ocurran problemas de este tipo.

Mejora en los servicios de copias de seguridad:

Muchos sistemas de ficheros dejan que sea el usuario quien proporcione las medidas necesarias para proteger los datos ante fallos en el sistema o en las aplicaciones. Los usuarios tienen que hacer copias de seguridad cada día, y si se produce algún fallo, utilizar estas copias para restaurarlos.

En este caso, todo el trabajo realizado sobre los datos desde que se hizo la última copia de seguridad se pierde y se tiene que volver a realizar. Sin embargo, los SGBD actuales funcionan de modo que se minimiza la cantidad de trabajo perdido cuando se produce un fallo.

También debemos mencionar las posibles desventajas de utilizar bases de datos

Complejidad:

Los SGBD son conjuntos de programas que pueden llegar a ser complejos con una gran funcionalidad. Es preciso comprender muy bien esta funcionalidad para poder realizar un buen uso de ellos.

Coste del equipamiento adicional:

Tanto el SGBD, como la propia base de datos, pueden hacer que sea necesario adquirir más espacio de almacenamiento. Además, para alcanzar las prestaciones deseadas, es posible que sea necesario adquirir una máquina más grande o una máquina que se dedique solamente al SGBD. Todo esto hará que la implantación de un sistema de bases de datos sea más cara.

Vulnerable a los fallos:

El hecho de que todo esté centralizado en el SGBD hace que el sistema sea más vulnerable ante los fallos que puedan producirse. Es por ello que deben tenerse copias de seguridad (Backup).

Veamos un ejemplo:

Supongamos que en una Base de Datos (de ahora en más BD) tenemos un listado de todos los teléfonos de las personas que viven en Capital Federal, y le hacemos las siguientes preguntas:

¿Cuáles son los teléfonos de todas las personas de apellido Gómez?

Como respuesta nos va a dar los teléfonos de todos los Gómez de Capital Federal que tenga ingresados.

¿Cuál es el nombre de la persona con el número de teléfono 0223-431-1343?

En este caso como respuesta obtendremos “vacío” ya que la característica indicada es de Mar del Plata, y esta BD sólo tiene disponible información sobre Capital Federal, por lo tanto no puede encontrar dato alguno que responda a nuestra consulta.

Estas preguntas a la BD, se realizan mediante un lenguaje llamado SQL (Structured Query Language – Lenguaje Estructurado de Consultas) y la BD nos va a responder con datos o “vacío” si es que no encontró ningún dato que respondiera a nuestra pregunta.

TIPOS DE BASES DE DATOS

Se utilizan para describir datos en los niveles conceptual y físico. Estos modelos utilizan registros e instancias para representar la realidad, así como las relaciones que existen entre estos registros (ligas) o apuntadores. A diferencia de los modelos de datos basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.

Entre las más utilizadas podemos mencionar las siguientes:

- Relacionales
- Orientadas a Objetos
- Lógicas

Si bien estos tres tipos de BD sirven para organizar la información y devolvernos respuestas adecuadas a nuestras preguntas, el único tipo que realmente se usa en las aplicaciones son las Bases de Datos Relacionales. La no utilización de los otros dos tipos se debe a cuestiones avanzadas de programación porque el desarrollo de programas con ellas es muy complicado.

Bases de Datos Relacionales

La ventaja del modelo relacional es que los datos se almacenan, al menos conceptualmente, de un modo en que los usuarios entienden con mayor facilidad. Los datos se almacenan como tablas y las relaciones entre las filas y las tablas son visibles en los datos. Este enfoque permite a los usuarios obtener información de la base de datos sin asistencia de sistemas profesionales de administración de información.

El siguiente ejemplo que desarrollaremos, muestra la aplicación de una base de datos relacional a un sistema de clientes, lo explicamos a continuación y lo graficamos más abajo:

Vemos que en la tabla de clientes muchos de sus campos se hallan codificados, por ejemplo, el vendedor está reemplazado en la tabla de clientes por un número, que luego se traduce, tabla de vendedores mediante, al nombre correspondiente que aparece en la ficha (recuadro grisado que vemos en la parte inferior de la figura). Algo similar ocurre con la provincia y la categoría de inscripción en el IVA.

El mecanismo de la relación es muy sencillo: basta con establecer la misma (indicando qué campos y qué tablas intervienen) para luego acceder automáticamente a los datos relacionados.

Por ejemplo, y con referencia a la figura, establecemos la relación del campo Cat. con la tabla de categorías IVA, del campo Vend. con la tabla de vendedores, etc.

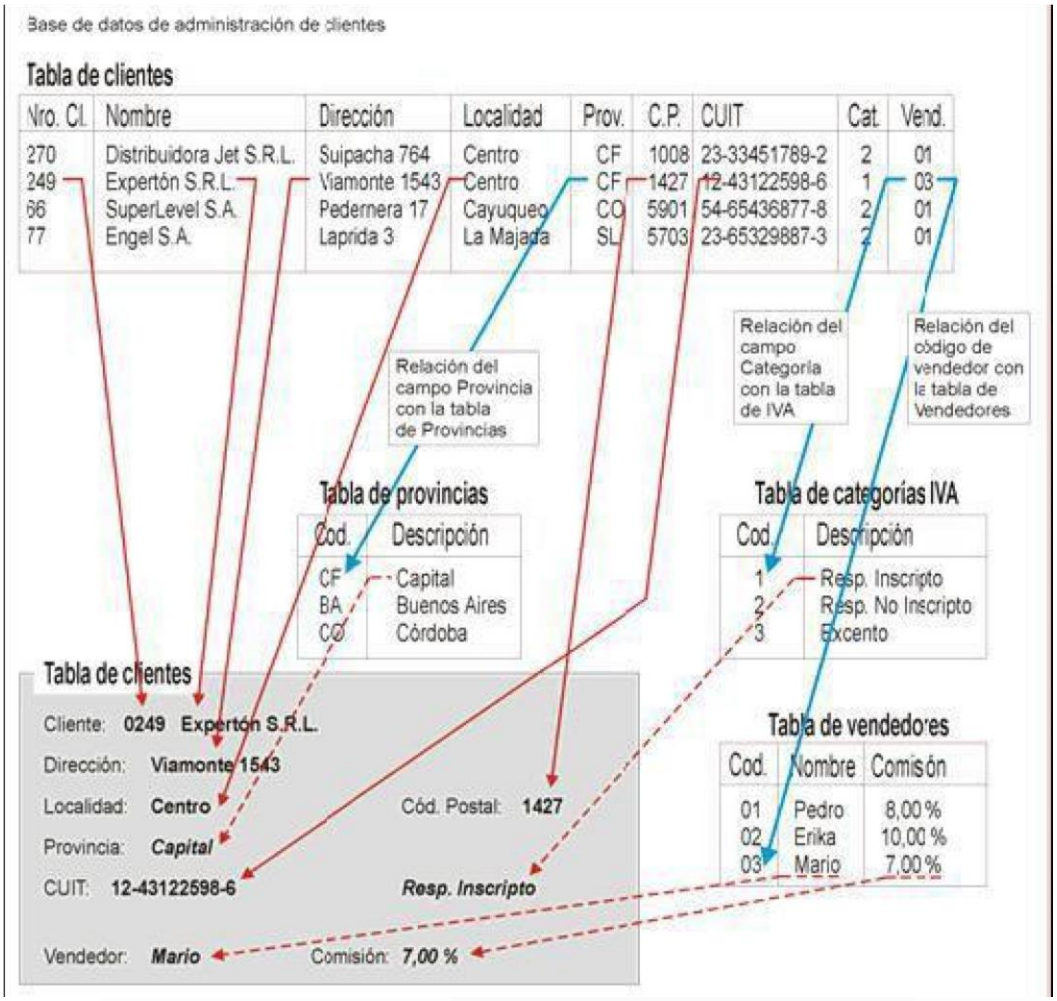
Una vez establecidas las mencionadas relaciones, cada vez que se lee un registro en la tabla de clientes, se posicionan en el lugar correspondiente todas las tablas que se encuentran relacionadas.

En el ejemplo, con sólo acceder al cliente número 249, la tabla vendedores (en los ejemplos amada

Comisión) se posicionará en "Mario" y la de categoría de IVA en "Resp. Inscripto".

Podemos mostrar entonces directamente el nombre del vendedor en lugar de su código, ocurriendo algo similar con la categoría IVA y la provincia. Elaboraremos así una vista del archivo de clientes con todas las descripciones necesarias. La información obtenida por medio de las relaciones se muestra en cursiva en el cuadro grisado de la figura, a este tipo de vista particular de la base lo llamaremos consulta.

Una base de datos relacional, finalmente, es un sistema específicamente diseñado para el manejo de información que ha sido previamente organizada en forma de una o varias tablas relacionadas entre sí.



Las características más importantes de los modelos relacionales son:

1. Es importante saber que las entradas en la tabla tienen un solo valor (son atómicos); no se admiten valores múltiples, por lo tanto la intersección de un renglón con una columna tiene un solo valor, nunca un conjunto de valores.
2. Todas las entradas de cualquier columna son de un solo tipo. Por ejemplo, una columna puede contener nombres de clientes, y en otra puede tener fechas de nacimiento. Cada columna posee un nombre único, el orden de las comunas no es de importancia para la tabla, las columnas de una tabla se conocen como atributos. Cada atributo tiene un dominio, que es una descripción física y lógica de valores permitidos.

3. No existen 2 filas en la tabla que sean idénticas.
4. La información en las bases de datos son representados como datos explícitos, no existen apuntadores o ligas entre las tablas.

En el enfoque relacional es sustancialmente distinto de otros enfoques en términos de sus estructuras lógicas y del modo de las operaciones de entrada/salida. En el enfoque relacional, los datos se organizan en tablas llamadas relaciones, cada una de las cuales se implanta como un archivo. En terminología relacional una fila en una relación representa un registro o una entidad; Cada columna en una relación representa un campo o un atributo.

Así, una relación se compone de una colección de entidades (o registros) cuyos propietarios están descritos por cierto número de atributos predeterminados implantados como campos.

En este modelo se representan los datos y las relaciones entre estos, a través de una colección de tablas, en las cuales los renglones equivalen a los cada uno de los registros que contendrá la base de datos y las columnas corresponden a las características (atributos o campos) de cada registro localizado.

Para resumir, en este tipo de bases de datos hay varios elementos que hay que tener en cuenta:

- Tablas
- Columnas o campos
- Filas o registros
- Valores
- Campos Llave
- Relaciones
- Esquemas

Tablas

Las bases relacionales se componen de relaciones, más comúnmente llamadas tablas.

Una tabla es lo que su nombre indica, un cuadro de doble entrada en el cual se relacionan datos. Es bastante parecido a cuando uno guarda información en una planilla de cálculo: allí también lo hace en tablas compuestas por Columnas y Filas.

Veamos la siguiente tabla, que muestra un ficticio ranking mundial de Tenis.

Ranking

Puesto	Nombre	Apellido	Edad	País
1	Roger	Federer	31	Suiza
2	Djokovic	Novak	25	Serbia
3	Rafael	Nadal	26	España
4	Andy	Murray	25	Reino Unido
5	David	Ferrer	30	España
6	Jo-Wilfried	Tsonga	27	Francia
7	Tomas	Berdych	26	República Checa
8	Juan Martín	Del Potro	23	Argentina
9	Janko	Tipsarevic	28	Serbia
10	John	Isner	27	Estados Unidos

La tabla tiene un nombre (Ranking), columnas que contienen distintos tipos de datos, y filas, que corresponden una a cada tenista ingresado.

Columnas o campos

Cada columna dentro de una tabla tiene un nombre único y almacenan diferente información cada una.

En nuestra tabla Ranking, no serviría de mucho tener dos columnas llamadas “apellido”, a no ser que las usemos para almacenar el primer y segundo apellido de los tenistas.

Pero en ese caso, deberíamos llamarlas con un nombre acorde a lo que contienen:

“primer_apellido” y “segundo_apellido”.

Además, cada Columna (también denominadas Campo) tiene asociado un Tipo de Dato.

Es lo mismo que las variables, si vamos a almacenar texto en un campo, tiene que ser un campo de tipo texto. En cambio si queremos almacenar un número, tendremos que usar un campo de tipo numérico.

Filas o registros

Cada fila en la tabla Ranking representa a un tenista diferente. Como las filas dependen de una tabla, todas van a tener la misma estructura pero contenidos diferentes entre sí.

Otra vez, no tiene ningún sentido tener dos filas de Federer en nuestra tabla porque tendrían exactamente los mismos datos.

A las filas se las suele llamar Registros.

Valores

Cada fila consiste de un grupo de valores, los cuales corresponden a cada una de las columnas. Como cada uno corresponde a una columna, su tipo de dato es el indicado por la columna en cuestión.

Puesto	Nombre	Apellido	Edad	País
1	Roger	Federer	31	Suiza
2	Djokovic	Novak	25	Serbia
3	Rafael	Nadal	26	España
4	Andy	Murray	25	Reino Unido
5	David	Ferrer	30	España
6	Jo-Wilfried	Tsonga	27	Francia

7	Tomas	Berdych	26	República Checa
8	Juan Martín	Del Potro	23	Argentina
9	Janko	Tipsarevic	28	Serbia
10	John	Isner	27	Estados Unidos

Columna (campo)
Fila (registro)
Valor

Llaves

Como dijimos antes, cada registro (o fila) tiene que tener información diferente, porque dos registros iguales serían generar datos de más en la BD. Ahora bien, tenemos que tener una manera de seleccionar solamente a un registro específico.

Veamos esto con la siguiente tabla:

nombre	apellido	direccion	telefono
Pablo	Martínez	Rivadavia 2564	4567-9852
Diego	Romar	Cucha Cucha 3506	4258-6985
Alfredo	Romero	Pasco 1151	4587-9856
Silvia	Lanzillotta	Crámer 1743	4358-5874
Adelina	Caraibo	Charcas 4312	4562-9874
Pablo	Martínez	San Pedrito 111	4578-4253
Diego	Gassi	Virrey Olaguer 2955	4312-9098
Jorge Darío	Paley	Juana Azurduy 1520	4313-4251
Nicolás	Moldavsky	Campichuelo 462	4125-7689

Si alguien nos preguntara si tenemos el teléfono del cliente Pablo Martínez, porque ocurrió un problema en la entrega de un producto en su domicilio y hay que contactarse con él, le preguntaríamos a la base de datos:

¿Cuál es el teléfono del cliente con nombre Pablo y apellido Martínez?

La respuesta sería una tabla con los siguientes datos:

nombre	apellido	direccion	telefono
Pablo	Martínez	Rivadavia 2564	4567-9852
Pablo	Martínez	San Pedrito 111	4578-4253

La respuesta que nos devolvió la BD es totalmente válida, los dos registros cumplen con la condición que le pedimos: el cliente Pablo Martínez. Si se fijan, en la tabla hay dos Pablo Martínez: Uno en la primera y otro en la sexta fila.

Para arreglar esto, tendríamos que preguntar cuál es la dirección del Pablo Martínez del que se necesita el teléfono y volver a generar la consulta, especificando un poco más:

¿Cuál es el teléfono del cliente con nombre Pablo, apellido Martínez y dirección Rivadavia 2564?

Ahora la respuesta sería una tabla con los siguientes datos:

nombre	apellido	direccion	telefono
Pablo	Martínez	Rivadavia 2564	4567-9852

Bien, logramos conseguir sólo el dato que nos hacía falta, pero tuvimos que usar muchas condiciones para lograr que la BD nos devolviera un registro único.

Es para evitar este tipo de situaciones que se usan los capos Llave: son campos (columnas) cuyo contenido va a ser único a lo largo de toda la tabla.

A fin de evitar que los valores que se vayan a almacenar en este tipo de campo se dupliquen, por lo general se usan campos numéricos que la base de datos maneja cuidándose siempre de asignar un número no usado anteriormente. Un claro ejemplo de esto en la vida real es el DNI, el número de la tarjeta de crédito, las cuentas bancarias, etc.

¿Cómo aplicamos esto a nuestra tabla de clientes?

Lo que podemos hacer es agregar un campo llave denominado “nro_cliente” que represente un Número Único de Cliente, el cual utilizaremos para referirnos unívocamente a cada uno de ellos. Entonces la tabla quedaría así:

Clientes

nro_cliente	nombre	apellido	dirección	teléfono
1	Pablo	Martínez	Rivadavia 2564	4567-9852
2	Diego	Romar	Cucha Cucha 3506	4258-6985
3	Alfredo	Romero	Pasco 1151	4587-9856
4	Silvia	Lanzillotta	Crámer 1743	4358-5874
5	Adelina	Caraibo	Charcas 4312	4562-9874
6	Pablo	Martínez	San Pedrito 111	4578-4253
7	Diego	Gassi	Virrey Olaguer 2955	4312-9098
8	Jorge Darío	Paley	Juana Azurduy 1520	4313-4251
9	Nicolás	Moldavsky	Campichuelo 462	4125-7689

En las Bases de Datos Relacionales, como su nombre lo indica, las relaciones entre las diversas tablas que la componen tienen un rol importante. Para indicar estas relaciones de una tabla a la otra, se utilizan los campos llave.

Veamos la siguiente tabla de Órdenes de Compra:

Compras

nro_pedido	nro_cliente	descripción	importe
1	5	50 Resmas A4	600
2	1	100 Lapiceras Negras	100
3	6	22 Cuadernos Oficio	110

Si analizamos la estructura de esta tabla, nos encontramos con dos campos llave:

nro_pedido y nro_cliente. Con “nro_pedido” lo que logramos es identificar unívocamente al pedido realizado, y con “nro_cliente”, podemos ubicar al cliente que realizó la compra. Si queremos saber quién compró “50 Resmas A4”, sólo hace falta ver el Número de Cliente que está en la columna “nro_cliente” y luego ir a la tabla Clientes para ver a quien le corresponde ese número, que en este caso es “Adelina Caraibo”.

Peligros en el diseño de bases de datos relacionales.

Uno de los retos en el diseño de la base de datos es el de obtener una estructura estable y lógica tal que:

- El sistema de base de datos no sufra de anomalías de almacenamiento.
- El modelo lógico pueda modificarse fácilmente para admitir nuevos requerimientos.

Una base de datos implantada sobre un modelo bien diseñado tiene mayor esperanza de vida aun en un ambiente dinámico, que una base de datos con un diseño pobre. En promedio, una base de datos experimenta una reorganización general cada seis años, dependiendo de lo dinámico de los requerimientos de los usuarios. Una base de datos bien diseñada tendrá un buen desempeño aunque aumente su tamaño, y será lo suficientemente flexible para incorporar nuevos requerimientos o características adicionales.

Existen diversos riesgos en el diseño de las bases de datos relacionales que afecten la funcionalidad de la misma, los riesgos generalmente son la redundancia de información y la inconsistencia de datos.

La normalización es el proceso de simplificar la relación entre los campos de un registro.

Por medio de la normalización un conjunto de datos en un registro se reemplaza por varios registros que son más simples y predecibles y, por lo tanto, más manejables. La normalización se lleva a cabo por cuatro razones:

- Estructurar los datos de forma que se puedan representar las relaciones pertinentes entre los datos.
- Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consultas y reportes.
- Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.
- Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.

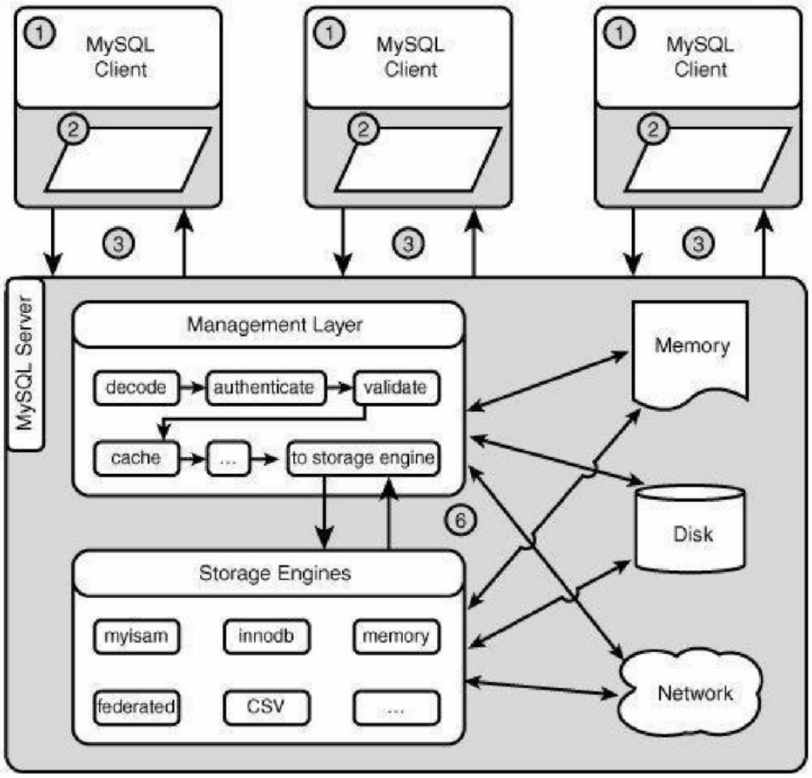
En términos más sencillos la normalización trata de simplificar el diseño de una base de datos, esto a través de la búsqueda de la mejor estructuración que pueda utilizarse con las entidades involucradas en ella.

La teoría de normalización tiene como fundamento el concepto de formas normales; se dice que una relación está en una determinada forma normal si satisface un conjunto de restricciones.

Dejemos de lado la generalidad de las bases de datos y comencemos a introducirnos en la base que utilizaremos durante el curso, es decir MySQL. Comencemos por entender cómo funciona entonces un servidor MySQL.

FUNCIONAMIENTO DE UN SERVIDOR MYSQL

1. Los clientes se conectan a servidor.
2. Los clientes inician autentificación, codifican y envían peticiones, comprimen y cifran peticiones, cachean los resultados del servidor.
3. El servidor procesa las peticiones y devuelve las respuestas.
4. Las peticiones son procesadas primero por la capa de manipulación, que las desencripta, valida su sintaxis, las busca en la caché, y las envía al correspondiente motor de almacenamiento.
5. Los motores de almacenamiento (MyISAM, InnoDB, Memory) manejan la representación en memoria y disco de bases de datos, tablas e índices, así como generación de estadísticas y algunos logs.
6. La capa de manejo escribe logs a disco, guarda y lee caches en memoria, lee logs binarios de la red. Los motores de almacenamiento guardan datos (tablas, logs, etc) en disco y en memoria, envía datos a otros servidores remotos.



- ***El servidor MySQL utiliza espacio en disco para almacenar lo siguiente:***

- Los programas cliente y servidor, y sus librerías. ○ Los ficheros de registro ("logs") y de estado. ○ Las bases de datos.
- Los ficheros de formato de tablas ('*.frm') para todos los motores de almacenamiento, y los ficheros de datos y ficheros de índices para algunos motores de almacenamiento.
- Los ficheros de "tablespaces" de InnoDB, si el motor de almacenamiento InnoDB está activado. ○ Tablas temporales internas que han sobrepasado el límite de tamaño en memoria y deben ser convertidas a tablas en disco.

- ***El servidor MySQL utiliza espacio en memoria para almacenar lo siguiente:***
  - Gestores de conexión (cada conexión consume memoria).
  - Buffers que guardan tablas temporales internas que no han sobrepasado el límite de tamaño en memoria.
  - Cachés: caché de hosts, la caché de tablas, la caché de consultas, etc. ○ Una copia de la tabla de permisos.
  - El contenido de las tablas HEAP (motor de almacenamiento en memoria). Su fichero de formato ('*.frm') se continua guardando en disco.

- ***El servidor MySQL utiliza los siguientes buffers por cada cliente:***
  - Buffers de registros para las búsquedas secuenciales en tablas ('read_buffer_size') y para leer las líneas después de una ordenación ('read_rnd_buffer_size') normalmente conseguida mediante la cláusula ORDER.
  - Buffer de join para las uniones de tablas.
  - Buffer de ordenación para las operaciones de ordenación.
  - Buffer de comunicaciones para intercambiar información con el cliente.

Comienza con un tamaño de 'net_buffer_length', pero si es necesario el servidor aumenta su tamaño al señalado por 'max_allowed_packet'.

- ***Los límites que el sistema operativo puede imponer al servidor MySQL son:***
  - El máximo número de ficheros abiertos por proceso limita el tamaño máximo de la caché de tablas, que guarda los descriptores de ficheros para los ficheros de tablas.
  - El máximo número de hilos de ejecución por proceso limita el número de clientes que se pueden conectar simultáneamente al servidor MySQL.

- El 'backlog' permitido por el sistema limita el número de conexiones de red en cola debido a clientes que esperan a conectarse.
- El sistema de ficheros donde se guardan los datos limita el tamaño máximo del fichero, pero este límite puede esquivarse repartiendo los datos en varios ficheros.

MOTORES DE ALMACENAMIENTO DE MYSQL Y TIPOS DE TABLAS

El servidor MySQL incorpora una característica única llamada "motores de almacenamiento", que nos permite seleccionar el tipo de almacenamiento interno de cada tabla, en base al que mejor se adecúe a una situación particular. Dicha selección, la hace el desarrollador a nivel de tabla, y no afecta a la manera en que el servidor interactúa con el cliente: los comandos SQL serán los mismos sea cual sea el motor de almacenamiento escogido. El cliente no necesita saber cómo se guardan los datos.

- Muy rápido en lectura y escritura (excepto escrituras simultáneas en la misma tabla). ○ Bajo requerimiento de espacio en disco y memoria.
- Los datos se guardan en disco: diferentes ficheros para la definición de la tabla, los datos y los índices.
- Es el motor por defecto de MySQL.
- Es una buena elección cuando necesitamos velocidad, y tenemos pocas modificaciones simultáneas de la tabla.

- ***Características del motor InnoDB:***
  - Transaccional.
  - Multiversionado: cuando múltiples transacciones modifican registros, InnoDB mantiene aisladas las transacciones guardando para cada una de ellas una versión distinta de un mismo registro, a cada transacción la versión que le corresponde.
  - Bloqueos a nivel de registro. ○ Restricciones en claves foráneas. ○ Fácil recuperación de datos en caso de error. ○ Alta concurrencia más segura en escritura. ○ Deshacer transacciones a medias ("rollback").
  - Los datos se guardan en disco: un fichero para la definición de la tabla, y un "tablespace" para guardar conjuntamente datos e índices. El tablespace puede consistir en uno o más ficheros, o incluso una partición entera en disco. ○ Podemos especificar cómo crecen los tablespaces en el fichero de configuración /etc/mysql/my.cnf.
  - Necesita más espacio en disco y memoria que MyISAM para guardar los datos (unas tres veces más de espacio en disco, y montones de RAM para las memorias temporales si queremos conseguir un rendimiento óptimo).
  - Es una buena elección cuando necesitamos transacciones, restricciones en claves foráneas, o tenemos muchas escrituras simultáneas.

- ***Características del motor HEAP (o MEMORY):***
  - Los datos se guardan en memoria, utilizando algoritmos que hacen un uso óptimo de este medio.
  - Es muy rápido. ○ Podemos crear una tabla HEAP a partir de una tabla en disco.
  - Es una buena elección cuando necesitamos realizar operaciones muy rápidas sobre conjuntos pequeños de datos.

- ***Características del motor NDB:***
  - Es el motor de almacenamiento de los clúster de MySQL.
  - La base de datos está repartida por los diferentes nodos del clúster. ○ Proporciona alta disponibilidad mediante redundancia.
  - Proporciona alto rendimiento mediante fragmentación de datos sobre los grupos de nodos.
  - Proporciona alta escalabilidad mediante la combinación de las dos características anteriores.
  - Los datos se guardan en memoria, pero los logs van a disco.
  - Es una buena elección cuando disponiendo de varios servidores necesitamos a la vez velocidad, transacciones y redundancia de datos; replicación síncrona; y resistencia a caídas de servidores.

LLAVES O CLAVES PRIMARIAS Y LLAVES O CLAVES FORÁNEAS

Dentro de los campos llave, podemos diferenciar dos tipos:

Primarias

Una clave primaria es un campo (o varios) que identifica unívocamente 1 solo registro (fila) en una tabla. Para un valor del campo clave existe solamente 1 registro. Los valores no se repiten ni pueden ser nulos.

Veamos un ejemplo, si tenemos una tabla con datos de personas, el número de documento puede establecerse como clave primaria, es un valor que no se repite; puede haber personas con igual apellido y nombre, incluso el mismo domicilio (padre e hijo por ejemplo), pero su documento será siempre distinto.

Si tenemos la tabla "usuarios", el nombre de cada usuario puede establecerse como clave primaria, es un valor que no se repite; puede haber usuarios con igual clave, pero su nombre de usuario será siempre distinto.

Si intentamos ingresar un valor para el campo clave que ya existe, aparece un mensaje de error indicando que el registro no se cargó pues el dato clave ya existe. Esto sucede porque los campos definidos como clave primaria no pueden repetirse.

Una tabla sólo puede tener una clave primaria. Cualquier campo (de cualquier tipo) puede ser clave primaria, debe cumplir como requisito, que sus valores no se repitan.

Al establecer una clave primaria estamos indexando la tabla, es decir, creando un índice para dicha tabla; a este tema lo veremos más adelante.

Foráneas

Son campos llave que permiten referenciar unívocamente a un registro que está dentro de otra tabla.

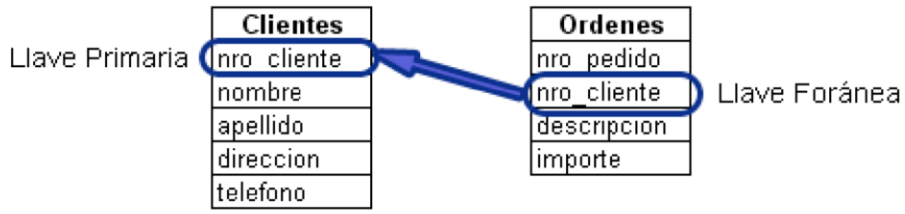
En el caso de la tabla “Ordenes”:

nro_pedido es una Llave Primaria

En este caso, el campo llave identifica a cada orden de compra, por lo tanto es para registros que están dentro de la tabla.

nro_cliente es una Llave Foránea

En cambio este campo se usa para saber qué cliente es el que realizó la orden de compra, es decir, nos sirve para identificar sin lugar a dudas el cliente de entre todos los de la tabla "Clientes".



Integridad referencial

La integridad referencial se refiere a las claves foráneas. Recordemos que una clave foránea es un atributo de una relación, cuyos valores se corresponden con los de una clave primaria en otra o en la misma relación. Este mecanismo se usa para establecer interrelaciones.

La integridad referencial consiste en que si un atributo o conjunto de atributos se define como una clave foránea, sus valores deben existir en la tabla en que ese atributo es clave principal.

Esquemas

El conjunto de todos los diseños de las tablas de una BD se llama Esquema. Vendría a ser algo así como un plano de la Base. En el Esquema de la BD se muestran todas las tablas, junto con sus campos y llaves primarias y/o foráneas.

Los esquemas pueden mostrarse mediante diagramas informales (como los que usamos arriba para indicar la relación entre Ordenes y Clientes), Diagramas de Entidad Relación, o mediante texto:

Clientes (*nro_cliente*, nombre, apellido, dirección, teléfono)

Ordenes (nro_pedido, *nro_cliente*, descripción, importe)

Con subrayado se indica la llave primaria y en *cursiva* la llave foránea.

TIPOS DE CAMPO

Diferentes tipos campos empleados en las bases de datos

Como sabemos una base de datos está compuesta de tablas donde almacenamos registros catalogados en función de distintos campos.

Un aspecto previo a considerar es la naturaleza de los valores que introducimos en esos campos. Dado que una base de datos trabaja con todo tipo de informaciones, es importante especificarle qué tipo de valor le estamos introduciendo de manera a, por un lado, facilitar la búsqueda posteriormente y por otro, optimizar los recursos de memoria.

Cada base de datos introduce tipos de valores de campo que no necesariamente están presentes en otras. Sin embargo, existe un conjunto de tipos que están representados en la totalidad de estas bases. Estos tipos comunes son los siguientes:

Alfanuméricos	Contienen cifras y letras. Presentan una longitud limitada (255 caracteres).
Numéricos	Existen de varios tipos, principalmente, enteros (sin decimales) y reales (con decimales).
Booleanos	Poseen dos formas: Verdadero y falso (Sí o No)
Fechas	Almacenan fechas facilitando posteriormente su explotación. Almacenar fechas de esta forma posibilita ordenar los registros por fechas o calcular los días entre una fecha y otra.
Memos	Son campos alfanuméricos de longitud ilimitada. Presentan el inconveniente de no poder ser indexados (veremos más adelante lo que esto quiere decir).
Autoincrementables	Son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. Su utilidad resulta más que evidente: Servir de identificador ya que resultan exclusivos de un registro.

Tipos de datos SQL

Los tipos de datos SQL se clasifican en 13 tipos de datos primarios y de varios sinónimos válidos reconocidos por dichos tipos de datos. Los tipos de datos primarios son:

Tipo de Datos	Longitud	Descripción
BINARY	1 byte	Para consultas sobre tabla adjunta de productos de bases de datos que definen un tipo de datos Binario.
BIT	1 byte	Valores Si/No ó True/False.
BYTE	1 byte	Un valor entero entre 0 y 255.
COUNTER	4 bytes	Un número incrementado automáticamente (de tipo Long).
CURRENCY	8 bytes	Un entero escalable entre 922.337.203.685.477,5808 y 922.337.203.685.477,5807.
DATETIME	8 bytes	Un valor de fecha u hora entre los años 100 y 9999
SINGLE	4 bytes	Un valor en punto flotante de precisión simple con un rango de - 3.402823*1038 a -1.401298*10-45 para valores negativos, 1.401298*10- 45 a 3.402823*1038 para valores positivos, y 0.
DOUBLE	8 bytes	Un valor en punto flotante de doble precisión con un rango de - 1.79769313486232*10308 a –

		4.94065645841247*10-324 para valores negativos, 4.94065645841247*10-324 a 1.79769313486232*10308 para valores positivos, y 0.
SHORT	2 bytes	Un entero corto entre -32,768 y 32,767.
LONG	4 bytes	Un entero largo entre -2,147,483,648 y 2,147,483,647.
LONGTEXT	1 byte por caracter.	De cero a un máximo de 1.2 gigabytes.
LONGBINARY	Según se necesite	De cero 1 gigabyte. Utilizado para objetos OLE.
TEXT	1 byte por caracter	De cero a 255 caracteres.

La siguiente tabla recoge los sinónimos de los tipos de datos definidos:

Tipo de Dato	Sinónimos
BINARY	VARBINARY
BIT	BOOLEAN LOGICAL LOGICAL1 YESNO
BYTE	INTEGER1
COUNTER	AUTOINCREMENT
CURRENCY	MONEY

DATETIME	DATE TIME TIMESTAMP
SINGLE	FLOAT4 IEEESINGLE REAL
DOUBLE	FLOAT FLOAT8 IEEEDOUBLE NUMBER NUMERIC
SHORT	INTEGER2 SMALLINT
LONG	INT INTEGER INTEGER4
LONGBINARY	GENERAL - OLEOBJECT
LONGTEXT	LONGCHAR MEMO NOTE
TEXT	ALPHANUMERIC CHAR – CHARACTER STRING – VARCHAR

UN VISTAZO AL SQL

Las aplicaciones en red son cada día más numerosas y versátiles. En muchos casos, el esquema básico de operación es una serie de scripts que rigen el comportamiento de una base de datos.

Debido a la diversidad de lenguajes y de bases de datos existentes, la manera de comunicar entre unos y otras sería realmente complicada a gestionar de no ser por la existencia de estándares que nos permiten el realizar las operaciones básicas de una forma universal.

Es de eso de lo que trata el Structured Query Language que no es más que un lenguaje estándar de comunicación con bases de datos. Hablamos por tanto de un lenguaje normalizado que nos permite trabajar con cualquier tipo de lenguaje (ASP o PHP) en combinación con cualquier tipo de base de datos (MS Access, SQL Server, MySQL...).

El hecho de que sea estándar no quiere decir que sea idéntico para cada base de datos.

En efecto, determinadas bases de datos implementan funciones específicas que no tienen necesariamente que funcionar en otras.

Aparte de esta universalidad, el SQL posee otras dos características muy apreciadas. Por una parte, presenta una potencia y versatilidad notables que contrasta, por otra, con su accesibilidad de aprendizaje.

MySQL es un sistema de gestión de bases de datos. Una base de datos define una estructura en la cual se guardará información. Los datos se guardan en tablas, en las cuales cada columna define el tipo de información que se guardará en ella y cada fila es un registro de la tabla. Cada base de datos tiene un nombre identificador, sus tablas tienen un nombre que las distingue y cada columna de una tabla tiene un tipo de dato asociado (por ejemplo, VARCHAR para cadenas de caracteres, INT para números enteros, etc).

El proceso de datos de las bases de datos se hace a través de consultas.

En las próximas unidades empezaremos a profundizar sobre este lenguaje.

Resumen

En esta Unidad…

En la presente unidad desarrollamos los conceptos necesarios comenzar a trabajar con bases de datos.

En la próxima Unidad…

En la próxima unidad vamos comenzar a trabajar con los parámetros de SQL necesarios para comenzar a interactuar con nuestras bases de datos, para prepararnos para incorporar contenido dinámico a nuestros sitios.