



DISEÑO DE BASE DE DATOS CON MYSQL

CONTENIDOS:

Formación de la comunidad de aprendizaje.....	4
Objetivo del curso.....	5
Historia de base de datos.....	6
Orígenes.....	7
Década de 1960.....	8
Década de 1970.....	9
Década de 1980.....	10
Década de 1990.....	10
Siglo XXI.....	11
Implementación.....	12
Sistemas de Gestión de Base de Datos (DBMS).....	12
Base de datos relacional.....	12
Ventajas de las bases de datos.....	12
Control sobre la redundancia de datos.....	12
Consistencia de datos.....	12
Compartir datos.....	12
Mantenimiento de estándares.....	13
Mejora en la integridad de datos.....	13
Mejora en la seguridad.....	13
Mejora en la accesibilidad a los datos.....	13
Mejora en la productividad.....	13
Mejora en el mantenimiento.....	13
Aumento de la concurrencia.....	13
Mejora en los servicios de copias de seguridad.....	14
Desventajas de las bases de datos.....	14
Complejidad.....	14
Coste del equipamiento adicional.....	14
Vulnerable a los fallos.....	14
Tipos de Campos.....	15
Fundamentos de base de datos.....	16

CONTENIDOS:

Fundamentos de la normalización.....	16
Primera forma normal	17
Segunda forma normal.....	17
Tercera forma normal.....	17
Otras formas de normalización.....	18
Entidades.....	21
¿Qué es el modelo entidad-relación?.....	21
Elementos del modelo entidad-relación.....	21
Relaciones de cardinalidad.....	24
CRUD.....	25
Base de datos relacionales.....	26
El modelo relacional.....	26
Modelo lógico.....	30
Que es SQL y cuáles son sus usos.....	32
Características principales.....	32
Sintaxis de SQL.....	33
MySQL.....	35
Instala MySQL.....	36
Generación y diseño de una base de datos en MySQL.....	41
Uso del Shell, comandos básicos y tablas.....	41
Conectándose a la base de datos.....	41
Manipulación avanzada de datos MySQL.....	50
Consultas condicionales y operaciones matemáticas en MySQL.....	50
Creación de la primera tabla.....	50
Copiar tablas.....	63
Vaciar y borrar tablas.....	64
Exportar tablas en PDF.....	69
Motores de la base de datos.....	70
Bibliografía/Cibergrafía.....	71

FORMACIÓN DE LA COMUNIDAD DE APRENDIZAJE:

La formación de la comunidad de aprendizaje es un proceso que debe llevarse a cabo para iniciar cada uno de nuestros cursos.

Su finalidad es crear un clima propicio para la celebración de la actividad instruccional, es decir, generar un entendimiento previo entre el instructor y los participantes sobre los temas que se desarrollarán durante ésta, así como las estrategias educativas que se llevarán a cabo para lograr un mejor aprendizaje.

Un adecuado manejo de la comunidad de aprendizaje es un elemento fundamental para garantizar la satisfacción de uno de los clientes involucrados en la impartición de los cursos: **los participantes**.

- **Presentación del Instructor:**

- Nombre, profesión, años de experiencia como instructor, experiencia en la impartición del curso, o cursos similares o relacionados.

- **Alineación de expectativas:**

- El instructor recabará las expectativas de los participantes respecto al curso, con el fin de dejarles claro el objetivo del mismo.

- En caso de que alguna expectativa no coincida con los temas que el curso contiene, el instructor dejará claro cuáles de las expectativas expresadas no serán cubiertas con el curso y porqué.

- Las expectativas alineadas serán anotadas en hojas de rotafolio para su revisión al término del curso.

- Durante el desarrollo del curso el instructor deberá cubrir las expectativas alineadas.

- **Presentación del objetivo del curso:** El instructor presentará a los participantes el objetivo del curso, aclarando dudas al respecto si las hubiese.

- **Reglas de oro:**

- El instructor promoverá el establecimiento de reglas por parte de los participantes que se observarán a través del curso; por lo que puede proponer: tiempo de tolerancia para iniciar las sesiones, respeto hacia los compañeros, participación de todos en técnicas y ejercicios grupales, etc.; se incluirán todos los puntos que los participantes consideren pertinentes.

- Se anotarán los acuerdos en hojas de rotafolio y se colocarán en un espacio en el que sean visibles a lo largo de todo el curso.

- **Cumplimiento de expectativas:** El instructor presentará a los participantes el objetivo del curso, aclarando dudas al respecto si las hubiese.

- Al finalizar el curso el instructor deberá llevar a cabo una revisión de las expectativas alineadas que se anotaron en hojas de rotafolio al inicio del curso

- Se revisará cada una de las expectativas alineadas palomeando las que hayan sido cumplidas, y el instructor explicará de qué manera se llevó a cabo tal cumplimiento.

OBJETIVO DEL CURSO:

Al finalizar el curso, los participantes podrán diseñar y administrar diversas bases de datos con la herramienta Open Source MySQL.

HISTORIA DE BASE DE DATOS:

El término bases de datos fue escuchado por primera vez en un simposio celebrado en California en 1963.

En una primera aproximación, se puede decir que *una base de datos es un conjunto de información relacionada que se encuentra agrupada o estructurada.*

Desde el punto de vista informático, *una 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.*

Por su parte, *un sistema de Gestión de Bases de datos es un tipo de software muy específico dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan; o lo que es lo mismo, una agrupación de programas que sirven para definir, construir y manipular una base de datos, permitiendo así almacenar y posteriormente acceder a los datos de forma rápida y estructurada.*

Actualmente, las bases de datos tienen un impacto decisivo sobre el creciente uso de las computadoras.

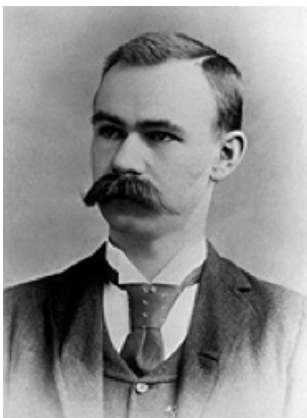
Pero para poder entender más profundamente una base de datos cabe entender su historia.

ORÍGENES:

Los orígenes de las bases de datos se remontan a la Antigüedad donde ya existían bibliotecas y toda clase de registros, se utilizaban para recoger información sobre las cosechas y censos.

Sin embargo, su búsqueda era lenta y poco eficaz y no se contaba con la ayuda de máquinas que pudiesen reemplazar el trabajo manual.

Posteriormente, el uso de las bases de datos se desarrolló por la necesidad de almacenar grandes cantidades de información o datos. Desde la aparición de las primeras computadoras, el concepto de bases de datos ha estado siempre ligado a la informática.



En 1884 *Herman Hollerith* creó la máquina automática de tarjetas perforadas, siendo nombrado así el *primer ingeniero estadístico de la historia*.

En esta época, los censos se realizaban de forma manual.

Ante esta situación, Hollerith comenzó a trabajar en el diseño de una máquina tabuladora o censadora, basada en tarjetas perforadas.

Posteriormente, en la década de los cincuenta se originan las cintas magnéticas, para automatizar la información y hacer respaldos.

Esto sirvió para suplir las necesidades de información de las nuevas industrias. Y a través de este mecanismo se empezaron a automatizar información, con la desventaja de que solo se podía hacer de forma secuencial.



DÉCADA DE 1960:

Posteriormente en la época de los sesenta, las computadoras bajaron los precios para que las compañías privadas las pudiesen adquirir; dando paso a que se popularizara el uso de los discos, cosa que fue un adelanto muy efectivo en la época, debido a que a partir de este soporte se podía consultar la información directamente, sin tener que saber la ubicación exacta de los datos.

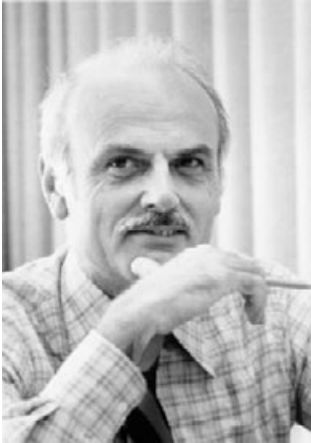
En esta misma época se dio inicio a las primeras generaciones de bases de datos de red y las bases de datos jerárquicas, ya que era posible guardar estructuras de datos en listas y árboles.

Otro de los principales logros de los años sesenta fue la alianza de IBM y American Airlines para desarrollar SABRE, un sistema operativo que manejaba las reservas de vuelos, transacciones e informaciones sobre los pasajeros de la compañía American Airlines.

Posteriormente, en esta década, se llevó a cabo el desarrollo del IDS desarrollado por Charles Bachman (qué formaba parte de la CODASYL) supuso la creación de un nuevo tipo de sistema de bases de datos conocido como modelo en red que permitió la creación de un standard en los sistemas de bases de datos gracias a la creación de nuevos lenguajes de sistemas de información.

CODASYL (*Conference on Data Systems Languages*) era un consorcio de industrias informáticas que tenían como objetivo la regularización de un lenguaje de programación estándar que pudiera ser utilizado en multitud de ordenadores.

Los miembros de este consorcio pertenecían a industrias e instituciones gubernamentales relacionadas con el proceso de datos, cuya principal meta era promover un análisis, diseño e implementación de los sistemas de datos más efectivos; y aunque trabajaron en varios lenguajes de programación como COBOL, nunca llegaron a establecer un estándar fijo, proceso que se llevó a cabo por ANSI.



ORACLE®

DÉCADA DE 1970:

Por lo que respecta a la década de los setenta, Edgar Frank Codd, científico informático inglés conocido por sus aportaciones a la teoría de bases de datos relacionales, definió el modelo relacional a la par que publicó una serie de reglas para los sistemas de datos relacionales a través de su artículo *“Un modelo relacional de datos para grandes bancos de datos compartidos”*

Este hecho dio paso al nacimiento de la segunda generación de los Sistemas Gestores de Bases de Datos.

Como consecuencia de esto, durante la década de 1970, Lawrence J. Ellison, más conocido como Larry Ellison, a partir del trabajo de Edgar F. Codd sobre los sistemas de bases de datos relacionales, desarrolló el Relational Software System, o lo que es lo mismo, lo que actualmente se conoce como Oracle Corporation, desarrollando así un sistema de gestión de bases de datos relacional con el mismo nombre que dicha compañía.

En la época de los ochenta también se desarrollará el SQL (*Structured Query Language*) o lo que es lo mismo un lenguaje de consultas o lenguaje declarativo de acceso a bases de datos relacionales que permite efectuar consultas con el fin de recuperar información de interés de una base de datos y hacer cambios sobre la base de datos de forma sencilla; además de analizar grandes cantidades de información y permitir especificar diversos tipos de operaciones frente a la misma información, a diferencia de las bases de datos de los ochenta, que se diseñaron para aplicaciones de procesamiento de transacciones.

Pero cabe destacar que ORACLE es considerado uno de los sistemas de bases de datos más completos que existen en el mundo, y aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace relativamente poco, actualmente sufre la competencia del SQL Server de la compañía Microsoft y de la oferta de otros Sistemas Administradores de Bases de Datos Relacionales con licencia libre como es el caso de PostgreSQL, MySQL o Firebird que aparecerían posteriormente en la década de 1990.



DÉCADA DE 1980:

Por su parte, a principios de los años ochenta comenzó el auge de la comercialización de los sistemas relacionales, y SQL comenzó a ser el estándar de la industria, ya que las bases de datos relacionales con su sistema de tablas (compuesta por filas y columnas) pudieron competir con las bases jerárquicas y de red, como consecuencia de que su nivel de programación era sencillo y su nivel de programación era relativamente bajo.

Microsoft



DÉCADA DE 1990:

En la década de 1990 la investigación en bases de datos giró en torno a las bases de datos orientadas a objetos, las cuales han tenido bastante éxito a la hora de gestionar datos complejos en los campos donde las bases de datos relacionales no han podido desarrollarse de forma eficiente. Así se desarrollaron herramientas como *Excel* y *Access* del paquete de *Microsoft Office* que marcan el inicio de las *bases de datos orientadas a objetos*.

Así se creó la *tercera generación de sistemas gestores de bases de datos*.

Fue también en esta época cuando se empezó a modificar la primera publicación hecha por ANSI del lenguaje SQL y se empezaron a agregar nuevas expresiones regulares, consultas recursivas, triggers y algunas características orientadas a objetos, que posteriormente en el siglo XXI volverá a sufrir modificaciones introduciendo características de XML, cambios en sus funciones, estandarización del objeto sequence y de las columnas autonuméricas.

Además, se creará la posibilidad de que SQL se pueda utilizar conjuntamente con XML, y se definirá las maneras de cómo importar y guardar datos XML en una base de datos SQL, dando así, la posibilidad de proporcionar facilidades que permiten a las aplicaciones integrar el uso de XQuery (*lenguaje de consulta XML*) para acceso concurrente a datos ordinarios SQL y documentos XML.

Posteriormente, se dará la posibilidad de usar la cláusula *order by*.

Aunque el boom de la década de los noventa será es *el nacimiento del World Wide Web* a finales de la década, ya que a través de este se facilitará la consulta a bases de datos.



SIGLO XXI:

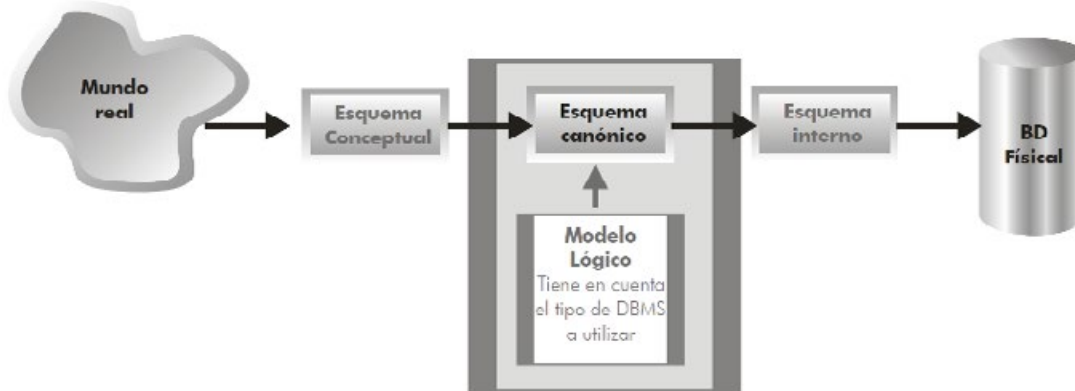
En la actualidad, las tres grandes compañías que dominan el mercado de las bases de datos son *IBM*, *Microsoft* y *Oracle*. Por su parte, en el campo de internet, la compañía que genera gran cantidad de información es *Google*.

Aunque existe una gran variedad de software que permiten crear y manejar bases de datos con gran facilidad, como por ejemplo *LINQ*, un proyecto de *Microsoft* que agrega consultas nativas semejantes a las de SQL a los lenguajes de la plataforma .NET.

El objetivo de este proyecto es permitir que todo el código hecho en Visual Studio sea también orientado a objetos; ya que antes de LINQ la manipulación de datos externos tenía un concepto más estructurado que orientado a objetos; es por eso que trata de facilitar y estandarizar el acceso a dichos objetos.

Cabe destacar que Visual Studio es un entorno de desarrollo integrado para sistemas operativos Windows que soporta varios lenguajes de programación tales como Visual C++, Visual#, Visual J#, ASP.NET y Visual Basic.NET, aunque se están desarrollando las extensiones necesarias para otros, cuyo objetivo es permitir crear aplicaciones, sitios y aplicaciones web, así como servicios web a cualquier entorno que soporte la plataforma .Net, creando así aplicaciones que intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

IMPLEMENTACIÓN:



SISTEMAS DE GESTIÓN DE BASE DE DATOS (DBMS)

BASE DE DATOS RELACIONAL:

Algunos ejemplos de SGBD son: *Oracle, MySQL, MS SQL Server*.

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.

VENTAJAS DE LAS BASES DE DATOS:

- 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.

- Compartir 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. 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.

DESVENTAJAS DE LAS 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).

TIPOS DE CAMPOS:

Cada Sistema de Base de Datos posee tipos de campos que pueden ser similares o diferentes. Entre los más comunes podemos nombrar:

- **Númérico:** Entre los diferentes tipos de campos numéricos podemos encontrar enteros *"sin decimales"* y reales *"decimales"*.
- **Booleanos:** Poseen dos estados: Verdadero *"Si"* y Falso *"No"*.
- **Memos:** Campos alfanuméricos de longitud ilimitada. Presentan el inconveniente de no poder ser indexados.
- **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.
- **Alfanuméricos:** contienen cifras y letras. Presentan una longitud limitada (255 caracteres).
- **Autoincrementables:** Campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. **Su utilidad resulta:** Servir de identificador ya que resultan exclusivos de un registro.

FUNDAMENTOS DE BASE DE DATOS

FUNDAMENTOS DE LA NORMALIZACIÓN:

La normalización es el proceso de organizar los datos de una base de datos. Se incluye la creación de tablas y el establecimiento de relaciones entre ellas según reglas diseñadas tanto para proteger los datos como para hacer que la base de datos sea más flexible al eliminar la redundancia y las dependencias incoherentes.

Los datos redundantes desperdician el espacio de disco y crean problemas de mantenimiento. Si hay que cambiar datos que existen en más de un lugar, se deben cambiar de la misma forma exactamente en todas sus ubicaciones. Un cambio en la dirección de un cliente es mucho más fácil de implementar si los datos sólo se almacenan en la tabla Clientes y no en algún otro lugar de la base de datos.

¿Qué es una "dependencia incoherente"? Aunque es intuitivo para un usuario mirar en la tabla Clientes para buscar la dirección de un cliente en particular, puede no tener sentido mirar allí el salario del empleado que llama a ese cliente. El salario del empleado está relacionado con el empleado, o depende de él, y por lo tanto se debería pasar a la tabla Empleados. Las dependencias incoherentes pueden dificultar el acceso porque la ruta para encontrar los datos puede no estar o estar interrumpida.

Hay algunas reglas en la normalización de una base de datos. Cada regla se denomina una "forma normal". Si se cumple la primera regla, se dice que la base de datos está en la "primera forma normal". Si se cumplen las tres primeras reglas, la base de datos se considera que está en la "tercera forma normal". Aunque son posibles otros niveles de normalización, la tercera forma normal se considera el máximo nivel necesario para la mayor parte de las aplicaciones.

Al igual que con otras muchas reglas y especificaciones formales, en los escenarios reales no siempre se cumplen los estándares de forma perfecta. En general, la normalización requiere tablas adicionales y algunos clientes consideran éste un trabajo considerable. Si decide infringir una de las tres primeras reglas de la normalización, asegúrese de que su aplicación se anticipa a los problemas que puedan aparecer, como la existencia de datos redundantes y de dependencias incoherentes.

PRIMERA FORMA NORMAL:

- Elimine los grupos repetidos de las tablas individuales.
- Cree una tabla independiente para cada conjunto de datos relacionados.
- Identifique cada conjunto de datos relacionados con una clave principal.

No use varios campos en una sola tabla para almacenar datos similares. Por ejemplo, para realizar el seguimiento de un elemento del inventario que proviene de dos orígenes posibles, un registro del inventario puede contener campos para el Código de proveedor 1 y para el Código de proveedor 2.

¿Qué ocurre cuando se agrega un tercer proveedor? Agregar un campo no es la respuesta, requiere modificaciones en las tablas y el programa, y no admite fácilmente un número variable de proveedores. En su lugar, coloque toda la información de los proveedores en una tabla independiente denominada Proveedores y después vincule el inventario a los proveedores con el número de elemento como clave, o los proveedores al inventario con el código de proveedor como clave.

SEGUNDA FORMA NORMAL:

- Cree tablas independientes para conjuntos de valores que se apliquen a varios registros.
- Relacione estas tablas con una clave externa.

Los registros no deben depender de nada que no sea una clave principal de una tabla, una clave compuesta si es necesario. Por ejemplo, considere la dirección de un cliente en un sistema de contabilidad. La dirección se necesita en la tabla Clientes, pero también en las tablas Pedidos, Envíos, Facturas, Cuentas por cobrar y Colecciones. En lugar de almacenar la dirección de un cliente como una entrada independiente en cada una de estas tablas, almacénela en un lugar, ya sea en la tabla Clientes o en una tabla Direcciones independiente.

TERCERA FORMA NORMAL:

- Elimine los campos que no dependan de la clave.

Los valores de un registro que no sean parte de la clave de ese registro no pertenecen a la tabla. En general, siempre que el contenido de un grupo de campos pueda aplicarse a más de un único registro de la tabla, considere colocar estos campos en una tabla independiente. Por ejemplo, en una tabla Contratación de empleados, puede incluirse el nombre de la universidad y la dirección de un candidato. Pero necesita una lista completa de universidades para enviar mensajes de correo electrónico en grupo. Si la información de las universidades se almacena en la tabla Candidatos, no hay forma de enumerar las universidades que no tengan candidatos en ese momento. Cree una tabla Universidades independiente y vincúlela a la tabla Candidatos con el código de universidad como clave.

- **EXCEPCIÓN:**

Cumplir la tercera forma normal, aunque en teoría es deseable, no siempre es práctico. Si tiene una tabla Clientes y desea eliminar todas las dependencias posibles entre los campos, debe crear tablas independientes para las ciudades, códigos postales, representantes de venta, clases de clientes y cualquier otro factor que pueda estar duplicado en varios registros. En teoría, la normalización merece el trabajo que supone. Sin embargo, muchas tablas pequeñas pueden degradar el rendimiento o superar la capacidad de memoria o de archivos abiertos. Puede ser más factible aplicar la tercera forma normal sólo a los datos que cambian con frecuencia. Si quedan algunos campos dependientes, diseñe la aplicación para que pida al usuario que compruebe todos los campos relacionados cuando cambie alguno..

OTRAS FORMAS DE NORMALIZACIÓN:

La cuarta forma normal, también llamada Forma normal de Boyce Codd (BCNF, Boyce Codd Normal Form), y la quinta forma normal existen, pero rara vez se consideran en un diseño real. Si no se aplican estas reglas, el diseño de la base de datos puede ser menos perfecto, pero no debería afectar a la funcionalidad.

NORMALIZAR UNA TABLA DE EJEMPLO:

Estos pasos demuestran el proceso de normalización de una tabla de alumnos ficticia.

- **Tabla sin normalizar:**

No. Alumno	Tutor	Despacho-Tut	Clase1	Clase2	Clase3
1022	García	412	101-07	143-01	159-02
4123	Díaz	216	201-01	211-02	214-01

- **Primera forma normal: no hay grupos repetidos**

Las tablas sólo deben tener dos dimensiones. Puesto que un alumno tiene varias clases, estas clases deben aparecer en una tabla independiente. Los campos Clase1, Clase2 y Clase3 de los registros anteriores son indicadores de un problema de diseño.

Las hojas de cálculo suelen usar la tercera dimensión, pero las tablas no deberían hacerlo. Otra forma de considerar ese problema es con una relación de uno a varios y poner el lado de uno y el lado de varios en tablas distintas. En su lugar, cree otra tabla en la primera forma normal eliminando el grupo repetido (Nº clase), según se muestra a continuación:

No. Alumno	Tutor	Despacho-Tut	No. clase
1022	García	412	101-07
1022	García	412	143-01
1022	García	412	159-02
4123	Díaz	216	201-01
4123	Díaz	216	211-02
4123	Díaz	216	214-01

• Segunda forma normal: eliminar los datos redundantes

Observe los diversos valores de N° clase para cada valor de N° alumno en la tabla anterior. N° clase no depende funcionalmente de N° alumno (la clave principal), de modo que la relación no cumple la segunda forma normal.

• Alumnos:

No. Alumno	Tutor	Despacho-Tut
1022	García	412
1022	García	412
1022	García	412
4123	Díaz	216
4123	Díaz	216
4123	Díaz	216

• Registro:

No. Alumno	No. clase
1022	101-07
1022	143-01
1022	159-02
4123	201-01
4123	211-02
4123	214-01

• Tercera forma normal: eliminar los datos no dependientes de la clave

En el último ejemplo, Despacho-Tut (el número de despacho del tutor) es funcionalmente dependiente del atributo Tutor. La solución es pasar ese atributo de la tabla Alumnos a la tabla Personal, según se muestra a continuación:

• Alumnos:

No. Alumno	Tutor
1022	García
4123	Díaz

• Personal:

Nombre	Habitación	Dept
García	412	42
Díaz	216	42

ENTIDADES

Las bases de datos son un gran pilar de la programación actual, ya que nos permiten almacenar y usar de forma rápida y eficiente cantidades ingentes de datos con cierta facilidad. En la actualidad se usa de forma mayoritaria las bases de datos relacionales (dominadas por distintos gestores a través del lenguaje SQL, en gran medida).

¿Qué es el modelo entidad-relación?

Como ya he comentado este modelo es solo y exclusivamente un método del que disponemos para diseñar estos esquemas que posteriormente debemos de implementar en un gestor de BBDD (bases de datos). Este modelo se representa a través de diagramas y está formado por varios elementos.

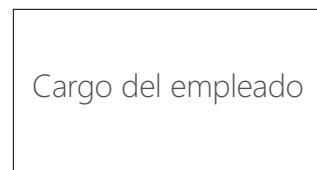
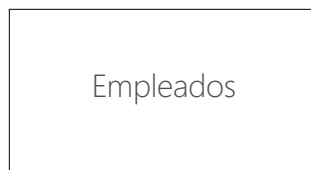
Este modelo habitualmente, además de disponer de un diagrama que ayuda a entender los datos y como se relacionan entre ellos, debe de ser completado con un pequeño resumen con la lista de los atributos y las relaciones de cada elemento.

ELEMENTOS DEL MODELO ENTIDAD-RELACIÓN

Entidad:

Las entidades representan cosas u objetos (ya sean reales o abstractos), que se diferencian claramente entre sí. Para poder seguir un ejemplo durante el artículo añadiré ejemplos sobre un taller mecánico, donde se podría crear las siguientes entidades:

- Coches (objeto físico): contiene la información de cada taller.
- Empleado (objeto físico): información de los trabajadores.
- Cargo del empleado (cosa abstracta): información de la función del empleado.



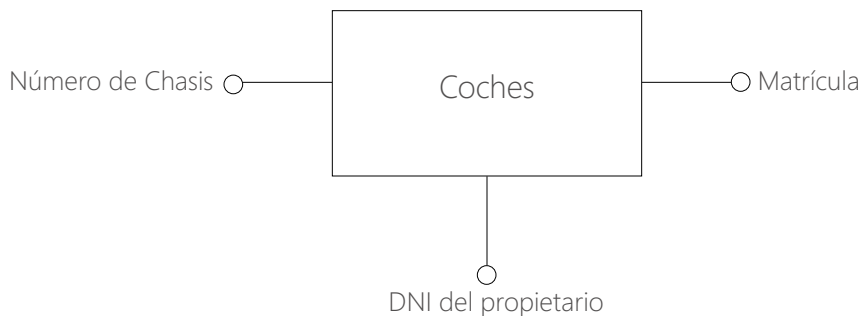
Atributos:

Los atributos definen o identifican las características de entidad (es el contenido de esta entidad). Cada entidad contiene distintos atributos, que dan información sobre esta entidad. Estos atributos pueden ser de distintos tipos (numéricos, texto, fecha...).

Siguiendo el ejemplo de antes podemos analizar los atributos de nuestra entidad "Coches", que nos darán información sobre los coches de nuestro supuesto taller.

Unos posibles atributos serían los siguientes: número de chasis, matrícula, DNI del propietario, marca, modelo y muchos otros que complementen la información de cada coche.

Los atributos se representan como círculos que descienden de una entidad, y no es necesario representarlos todos, sino los más significativos, como a continuación.



En un modelo relacional (ya implementado en una base de datos) un ejemplo de tabla dentro de una BBDD podría ser el siguiente.

Número de Chasis	Matrícula	DNI del propietario
5tfem5f10ax007210	4817 BFK	45338600L
6hsen2j98as001982	8810 CLM	02405068K
5rgsb7a19js001982	0019 GGL	40588860J

Este ejemplo es con tres atributos, pero un coche podría tener cientos (si fuese necesario) y seguirían la misma estructura de columnas, tras implementarlo en una BBDD.

Relación:

Es un vínculo que nos permite definir una dependencia entre varias entidades, es decir, nos permite exigir que varias entidades compartan ciertos atributos de forma indispensable.

Por ejemplo, los empleados del taller (de la entidad "Empleados") tienen un cargo (según la entidad "Cargo del empleado"). Es decir, un atributo de la entidad "Empleados" especificará que cargo tiene en el taller, y tiene que ser idéntico al que ya existe en la entidad "Cargo del empleado".

Las relaciones se muestran en los diagramas como rombos, que se unen a las entidades mediante líneas.



Yo, bajo mi punto de vista, entiendo mejor esto en una tabla (de una implementación en una BBDD), por lo que voy a poner el ejemplo de cómo se representaría (resaltada la relación, que posteriormente veremos cómo se haría).

● Empleados:

Nombre	DNI	Cargo
Carlos Sánchez	45338600L	001
Pepe Sánchez	02405068K	002
Juan Sánchez	40588860J	002

● Cargo del empleado:

ID del cargo	Descripción
001	Jefe de taller
002	Mecánico

Relaciones de cardinalidad:

Podemos encontrar distintos tipos de relaciones según como participen en ellas las entidades.

Es decir, en el caso anterior cada empleado puede tener un cargo, pero un mismo cargo lo pueden compartir varios empleados.

Esto complementa a las representaciones de las relaciones, mediante un intervalo en cada extremo de la relación que especifica cuantos objetos o cosas (de cada entidad) pueden intervenir en esa relación.

- **UNO A UNO:** Una entidad se relaciona únicamente con otra y viceversa. Por ejemplo, si tuviésemos una entidad con distintos chasis y otra con matrículas deberíamos de determinar que cada chasis solo puede tener una matrícula (y cada matrícula un chasis, ni más en ningún caso)



- **UNO A VARIOS O VARIOS A UNO:** determina que un registro de una entidad puede estar relacionado con varios de otra entidad, pero en esta entidad existir solo una vez. Como ha sido en el caso anterior del trabajador del taller.



- **VARIOS A VARIOS:** determina que una entidad puede relacionarse con otra con ninguno o varios registros y viceversa. Por ejemplo, en el taller un coche puede ser reparado por varios mecánicos distintos y esos mecánicos pueden reparar varios coches distintos.



Los indicadores numéricos indican el primero el número mínimo de registros en una relación y posteriormente el máximo (si no hay límite se representa con una "n").

Claves:

Es el atributo de una entidad, al que le aplicamos una restricción que lo distingue de los demás registros (no permitiendo que el atributo específico se repita en la entidad) o le aplica un vínculo (exactamente como comentábamos en las relaciones). Estos son los distintos tipos:

Superclave: aplica una clave o restricción a varios atributos de la entidad, para así asegurarse que en su conjunto no se repitan varias veces y así no poder entrar en dudas al querer identificar un registro.

Clave primaria: identifica inequívocamente un solo atributo no permitiendo que se repita en la misma entidad. Como sería la matrícula o el número de chasis de un coche (no puede existir dos veces el mismo).

Clave externa o clave foránea: este campo tiene que estar estrictamente relacionado con la clave primaria de otra entidad, para así exigir que exista previamente ese clave. Anteriormente hemos hablado de ello cuando comentábamos que un empleado indispensablemente tiene que tener un cargo (que lo hemos representado numéricamente), por lo cual si intentásemos darle un cargo inexistente el gestor de bases de datos nos devolvería un error.

CRUD

Toda aplicación web dispone de una importante cantidad de información (usuarios, contraseñas, fotos, localizaciones) en forma de datos que hay que almacenar de alguna manera (base de datos) para que podamos acceder a ellos e interactuar cuando lo necesitemos.

Un **CRUD** puede considerarse como uno de los principios básicos de cualquier aplicación web.

CRUD (Create, Read, Update, Delete) Es un método de programación que permite realizar las acciones de insertar, mostrar, modificar y borrar elementos de una base de datos.

BASE DE DATOS RELACIONALES:

El diseño de una base de datos consiste en definir la estructura de los datos que debe tener un sistema de información determinado. Para ello se suelen seguir por regla general unas fases en el proceso de diseño, definiendo para ello el modelo conceptual, el lógico y el físico.

- En el diseño conceptual se hace una descripción de alto nivel de la estructura de la base de datos, independientemente del SGBD (*Sistema Gestor de Bases de Datos*) que se vaya a utilizar para manipularla. Su objetivo es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar dicha información.
- El diseño lógico parte del resultado del diseño conceptual y da como resultado una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD.

El diseño lógico depende del tipo de SGBD que se vaya a utilizar, se adapta a la tecnología que se debe emplear, pero no depende del producto concreto. En el caso de bases de datos convencionales relacionales (*basadas en SQL para entendernos*), el diseño lógico consiste en definir las tablas que existirán, las relaciones entre ellas, normalizarlas, etc...

- El diseño físico parte del lógico y da como resultado una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Aquí el objetivo es conseguir una mayor eficiencia, y se tienen en cuenta aspectos concretos del SGBD sobre el que se vaya a implementar. Por regla general esto es transparente para el usuario, aunque conocer cómo se implementa ayuda a optimizar el rendimiento y la escalabilidad del sistema.

EL MODELO RELACIONAL:

En el modelo relacional las dos capas de diseño conceptual y lógico, se parecen mucho. Generalmente se implementan mediante diagramas de Entidad/Relación (*modelo conceptual*) y tablas y relaciones entre éstas (*modelo lógico*). Este es el modelo utilizado por los sistemas gestores de datos más habituales (*SQL Server, Oracle, MySQL...*).

Nota: Aunque mucha gente no lo sabe, a las bases de datos relaciones se les denomina así porque almacenan los datos en forma de "Relaciones" o listas de datos, es decir, en lo que llamamos habitualmente "Tablas". Muchas personas se piensan que el nombre viene porque además las tablas se relacionan entre sí utilizando claves externas. No es así, y es un concepto que debemos tener claro. (Tabla = Relación).

El modelo relacional de bases de datos se rige por algunas normas sencillas:

- Todos los datos se representan en forma de tablas (también llamadas “relaciones”, ver nota anterior). Incluso los resultados de consultar otras tablas. La tabla es además la unidad de almacenamiento principal.
- Las tablas están compuestas por filas (o registros) y columnas (o campos) que almacenan cada uno de los registros (*la información sobre una entidad concreta, considerados una unidad*).
- Las filas y las columnas, en principio, carecen de orden a la hora de ser almacenadas. Aunque en la implementación del diseño físico de cada SGBD esto no suele ser así, por ejemplo, en SQL Server si añadimos una clave de tipo “Clustered” a una tabla haremos que los datos se ordenen físicamente por el campo correspondiente.
- El orden de las columnas lo determina cada consulta (*que se realizan usando SQL*).
- Cada tabla debe poseer una clave primaria, esto es, un identificador único de cada registro compuesto por una o más columnas.
- Para establecer una relación entre dos tablas es necesario incluir, en forma de columna, en una de ellas la clave primaria de la otra. A esta columna se le llama clave externa. Ambos conceptos de clave son extremadamente importantes en el diseño de bases de datos.

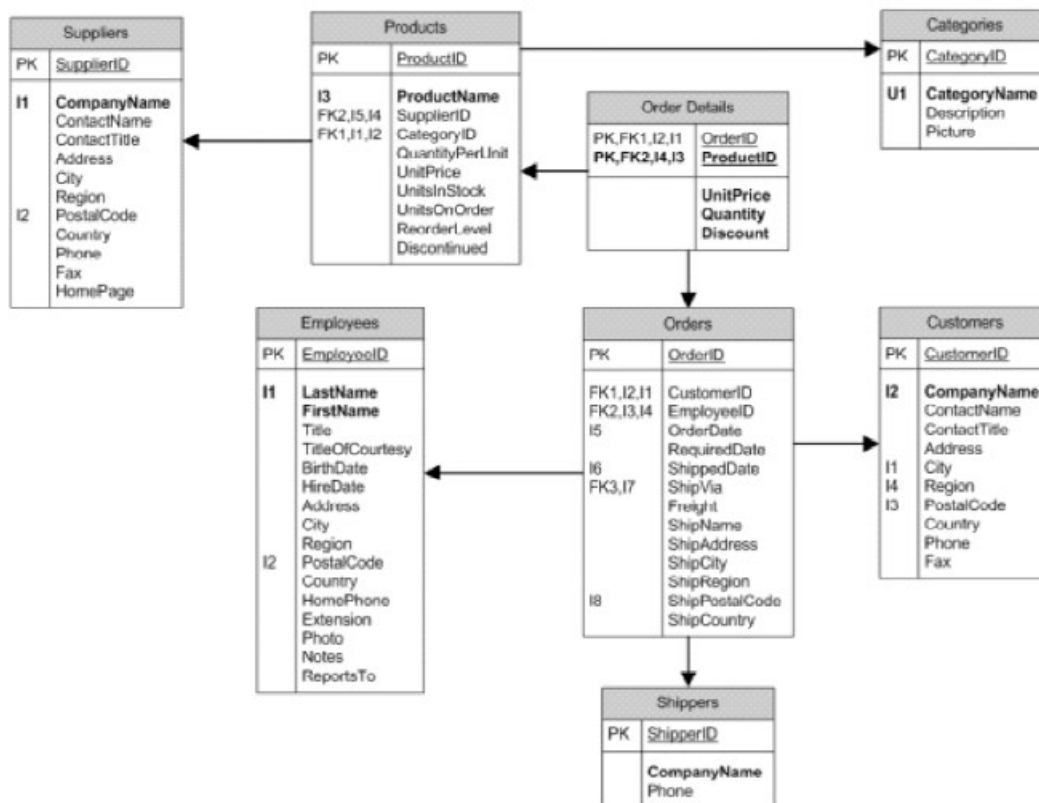
Basándose en estos principios se diseñan las diferentes bases de datos relacionales, definiendo un diseño conceptual y un diseño lógico, que luego se implementa en el diseño físico usando para ello el gestor de bases de datos de nuestra elección (*por ejemplo, SQL Server*).

Por ejemplo, consideremos la conocida base de datos Northwind de Microsoft.

Esta base de datos representa un sistema sencillo de gestión de pedidos para una empresa ficticia.

Existen conceptos que hay que manejar como: proveedores, empleados, clientes, empresas de transporte, regiones geográficas, y por supuesto pedidos y productos.

El diseño conceptual de la base de datos para manejar toda esta información se puede ver en la siguiente figura, denominada diagrama Entidad/Relación o simplemente diagrama E-R:



Como vemos existen tablas para representar cada una de estas entidades del mundo real: Proveedores (*Suppliers*), Productos, Categorías de productos, Empleados, Clientes, Transportistas (*Shippers*), y Pedidos (*Orders*) con sus correspondientes líneas de detalle (*Order Details*).

Además, están relacionadas entre ellas de modo que, por ejemplo, un producto pertenece a una determinada categoría (*se relacionan por el campo CategoryID*) y un proveedor (*SupplierID*), y lo mismo con las demás tablas.

Cada tabla posee una serie de campos que representan valores que queremos almacenar para cada entidad.

Por ejemplo, un producto posee los siguientes atributos que se traducen en los campos correspondientes para almacenar su información: Nombre (*ProductName*), Proveedor (*SupplierID*, que identifica al proveedor), Categoría a la que pertenece (*CategoryID*), Cantidad de producto por cada unidad a la venta (*QuantityPerUnit*), Precio unitario (*UnitPrice*), Unidades que quedan en stock (*UnitsInStock*), Unidades de ese producto que están actualmente en pedidos (*UnitsOnOrder*), qué cantidad debe haber para que se vuelva a solicitar más producto al proveedor (*ReorderLevel*) y si está descatalogado o no (*Discontinued*).

Los campos marcados con "PK" indican aquellos que son claves primarias, es decir, que identifican de manera única a cada entidad. Por ejemplo, ProductID es el identificador único del producto, que será por regla general un número entero que se va incrementando cada vez que introducimos un nuevo producto (1, 2, 3, etc..).

Los campos marcados como "FK" son claves foráneas o claves externas. Indican campos que van a almacenar claves primarias de otras tablas de modo que se puedan relacionar con la tabla actual. Por ejemplo, en la tabla de productos el campo CategoryID está marcado como "FK" porque en él se guardará el identificador único de la categoría asociada al producto actual. En otras palabras: ese campo almacenará el valor de la clave primaria (PK) de la tabla de categorías que identifica a la categoría en la que está ese producto.

Los campos marcados con indicadores que empiezan por "I" (ej: "I1") se refieren a índices. Los índices generan información adicional para facilitar la localización más rápida de registros basándose en esos campos.

Por ejemplo, en la tabla de empleados (*Employees*) existe un índice "I1" del que forman parte los campos Nombre y Apellidos (*en negrita además porque serán también valores únicos*) y que indica que se va a facilitar la locación de los clientes mediante esos datos. También tiene otro índice "I2" en el campo del código postal para localizar más rápidamente a todos los clientes de una determinada zona.

Los campos marcados con indicadores que empiezan con "U" (*por ejemplo U1*) se refieren a campo que deben ser únicos. Por ejemplo, en la tabla de categorías el nombre de ésta (*CategoryName*) debe ser único, es decir, no puede haber -lógicamente- dos categorías con el mismo nombre.

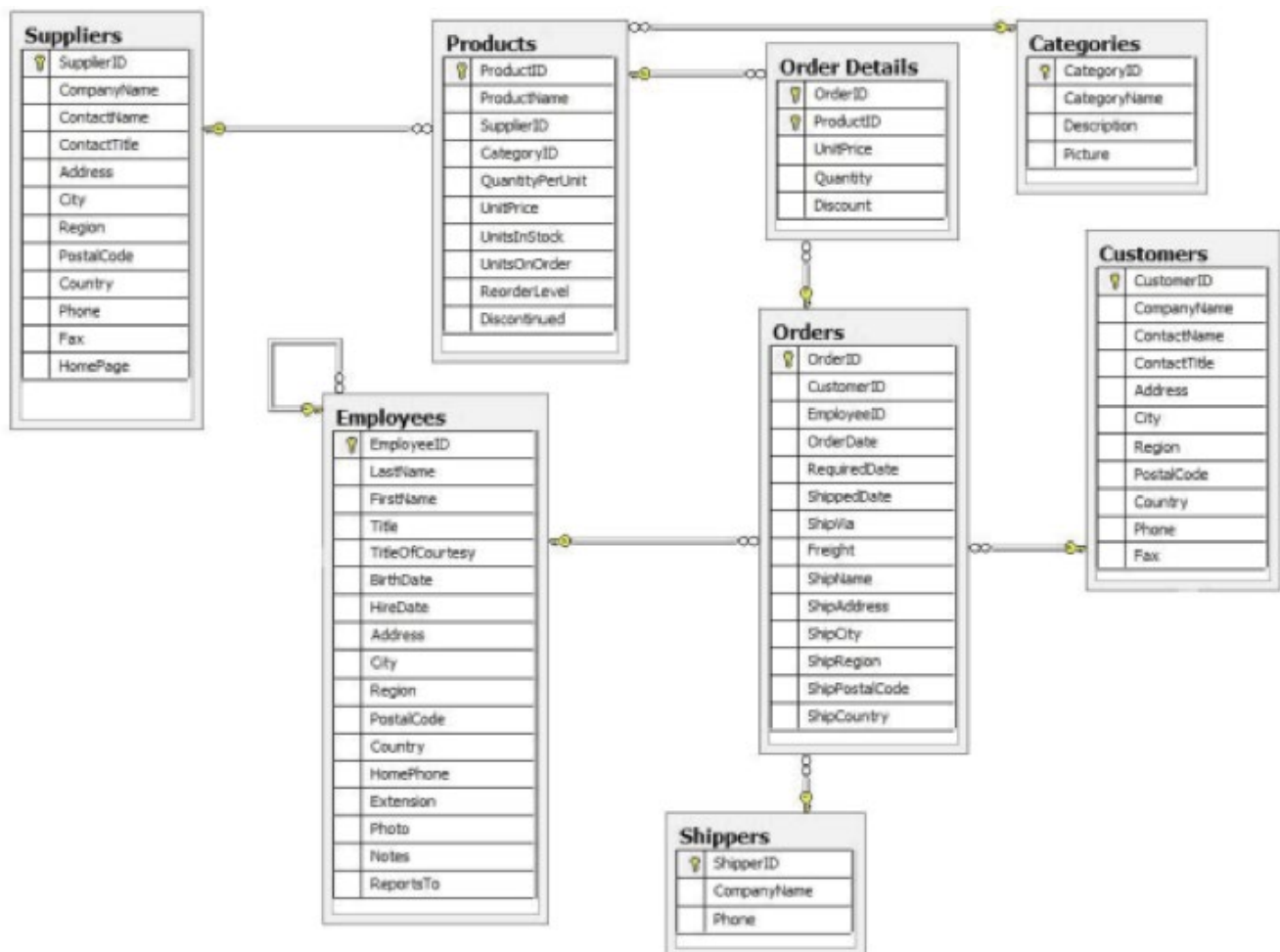
Como vemos, un diseño conceptual no es más que una representación formal y acotada de entidades que existen en el mundo real, así como de sus restricciones, y que están relacionadas con el dominio del problema que queremos resolver.

MODELO LÓGICO

Una vez tenemos claro el modelo E-R debemos traducirlo a un modelo lógico directamente en el propio sistema gestor de bases de datos (Oracle, MySQL, SQL Server...). Si hemos utilizado alguna herramienta profesional para crear el diagrama E-R, seguramente podremos generar automáticamente las instrucciones necesarias para crear la base de datos.

La mayoría de los generadores de diagramas E-R (*por ejemplo, Microsoft Visio*) ofrecen la capacidad de exportar el modelo directamente a los SGBD más populares.

Entonces, todo este modelo conceptual se traduce en un modelo lógico que trasladaremos a la base de datos concreta que estemos utilizando y que generalmente será muy parecido.
Por ejemplo, este es el mismo modelo anterior, mostrado ya como tablas en un diagrama de SQL Server:



En este caso hemos creado cada tabla, una a una, siguiendo lo identificado en el diagrama E-R y estableciendo índices y demás elementos según las indicaciones de cada uno de los campos, además, hemos decidido el mejor tipo de datos que podemos aplicar a cada campo (*texto, números, fechas... que se almacenan para cada registro*).

Su representación gráfica en la base de datos es muy similar, sin embargo, el modelo físico (*cómo se almacena esto físicamente*), puede variar mucho de un SGBD a otro y según la configuración que le demos.

QUÉ ES SQL Y CUÁLES SON SUS USOS

La sigla SQL significa **Structured Query Language**, o su equivalente en Español **Lenguaje de Pregunta Estructurado**, Este es un lenguaje Universal que está implementado en todos los Motores de Bases de Datos razón por la cual el SQL es el lenguaje estándar de comunicación entre los diferentes Motores existentes.

La creación de este lenguaje es sin duda alguna uno de los más importantes avances en el mundo de las bases de datos, si este no existiera, el tiempo que tomaría pasar información de un MBD a otro, sería realmente extenso y haría de los MBD algo complicado.

SQL es un lenguaje completamente normalizado que facilita el trabajo con cualquier tipo de lenguaje a la par con cualquier tipo de Base de Datos, sin embargo, esto no es equivalente a decir que es igual en todos los MBD, estos implementan diferentes funciones de acuerdo a la manera como más favorezca al MBD, estas funciones no siempre funcionan en otros.

CARACTERÍSTICAS PRINCIPALES:

- Lenguaje que permite el acceso a las bases de datos
- Aprovecha al máximo el poder y la flexibilidad de los Sistemas Relacionales, lo cual facilita las operaciones necesarias sobre estos.
- Es un lenguaje declarativo de alto nivel
- Permite una elevada productividad en codificación gracias a su base teórica.
- Aparte de la facilidad para efectuar consultas, SQL también puede servir como LDD, LDV y LMD
- Permite concesión y negación de permisos, restricciones de integridad, controles a la transacción y modificación de los esquemas.
- El lenguaje fue modificado con el fin de mantenerlo solo a nivel conceptual y externo
- Se puede usar de manera Interactiva, para esto, las sentencias SQL se escriben y se llevan acabo en líneas de comandos
- También puede usarse de manera Integrada, que está dirigido a usuarios mas avanzados, que utilizan un lenguaje de programación anfitrión y el SQL como sublenguaje de datos
- El SQL Estático es una técnica para el manejo embebido del SQL, y las sentencias que se utilizan no varían en ningún momento mientras se lleve a cabo la ejecución del programa.
- El SQL Dinámico también es una técnica para el uso embebido del SQL, pero a diferencia del SQL estático, esta modifica todas o gran parte de las sentencias mientras se ejecuta el programas
- Para evitar problemas en el orden de ejecución interno, se debe llevar a cabo una optimización, antes de ejecutar las sentencias.
- Otra versión del SQL es el FSQL, que es el mismo SQL, pero basado en lógica difusa, para lógicamente ser implementado en bases de datos difusas.

SINTAXIS DE SQL

Comandos para definición de datos:

- **CREATE TABLE:** Se utiliza para crear una nueva relación a la que se le asigna un nombre y unos atributos:
- **DROP TABLE:** Borra una relación existente así como también sus atributos y la tupla asignada a esta relación
- **ALTER TABLE:** Modifica la tabla, agrega un atributo a una de estas, además de cambiar la tupla del código de la Base de Datos
- **CREATE INDEX:** Comando empleado para crear índices, estos índices se crean bajo un nombre y pueden ser eliminados cuando son innecesarios
- **DROP INDEX:** Este comando es usado para borrar los índices de la tabla relacionada y la tupla del catálogo.

Comandos para manipulación de datos:

- **SELECT:** Esta instrucción tienen como fin, recuperar la información desde una base de datos. Existen funciones que están relacionadas con el comando SELECT, por ejemplo:
- **DISTINCT:** Antes de ejecutar la sentencia SELECT, esta instrucción borrara todos los errores de redundancia de datos que puedan existir.
- **COUNT:** Se utiliza para obtener el número de valores en la columna
- **SUM:** Suma todos los elementos de una columna, siempre y cuando estos sean numéricos
- **AVG:** Hace un promedio de los datos numéricos de una columna
- **MIN o MAX:** Se usa para obtener el mayor o menor valor de una Columna
- **COUNT(*):** Se implementa para contar la orientación de una tabla sin eliminación de valores duplicados
- **GROUP BY:** Reordena virtual, lógicamente y en grupos una tabla
- **HAVING:** Esta sentencia se usa para eliminar grupos de datos
- **ORDER BY:** Ordena la tabla en un orden específico
- **EXIST:** Esta función es una especie de calificador de existencia, es decir, evalúa todos los precoseos lógicos y se cumple cuando el retorno de estos no son nulos

Una subconsulta se hace combinando el Parámetro SELECT con cualquiera de las anteriores Instrucciones.

- **UPDATE:** Se utiliza para modificar los atributos de una o mas tuplas seleccionadas
- **DELETE:** Comando utilizado para borrar las tuplas desde una relación, si se digita solo, se borran todas, pero al combinarlo con el comando WHERE, se pueden seleccionar las tuplas que se van a borrar
- **INSERT:** Agrega una tupla a una relación, para esto se debe especificar el nombre de la relación y una lista ordenada de valores que se agregaran a la tupla.

MYSQL

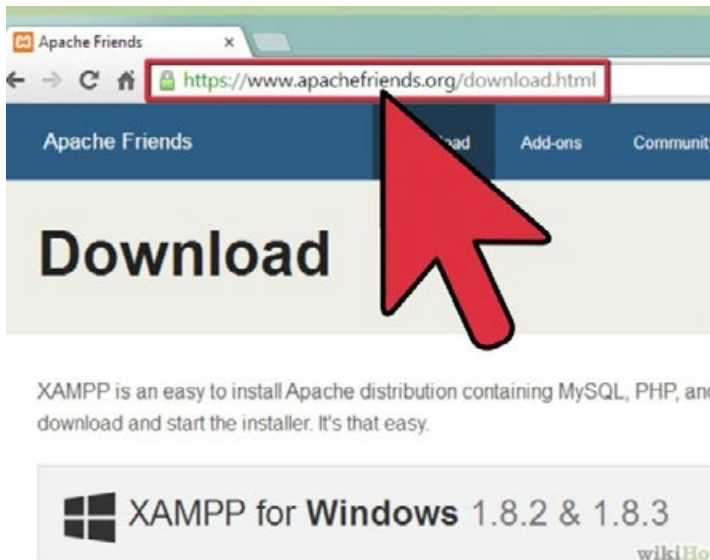
MySQL es un sistema de administración de bases de datos (*Database Management System, DBMS*) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interacción con los lenguajes de programación más utilizados como PHP, Perl y Java, y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

INSTALA MYSQL

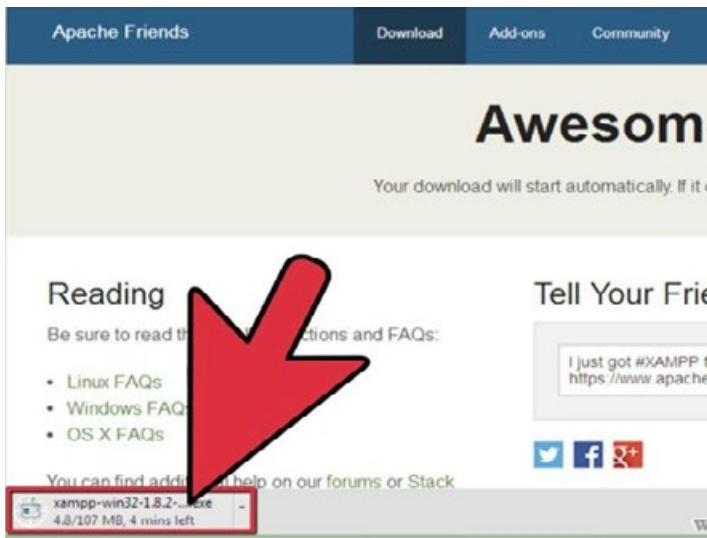
Cómo instalar un servidor de MySQL en una PC



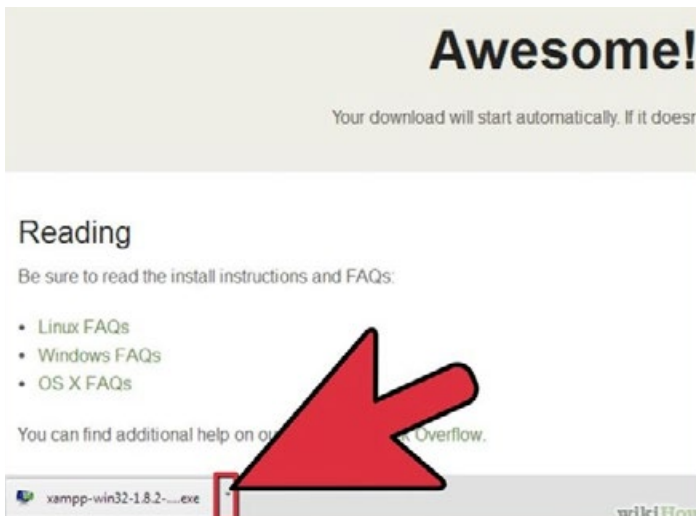
1: En tu navegador web, ve a:
<http://www.apachefriends.org/en/xampp-windows.html>.



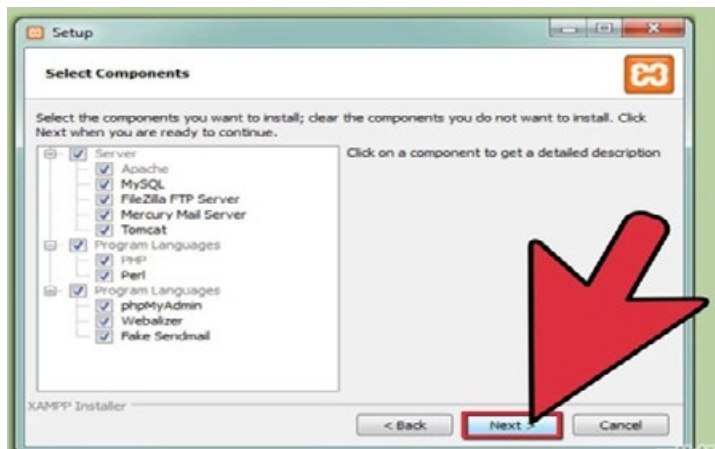
2: Haz clic en el enlace de descarga para descargar XAMPP.



3: Cuando aparezca la ventana de descarga, haz clic en "Guardar" y espera a que la descarga finalice.



4: Una vez que tu descarga termine, instala el programa haciendo clic en "Ejecutar".

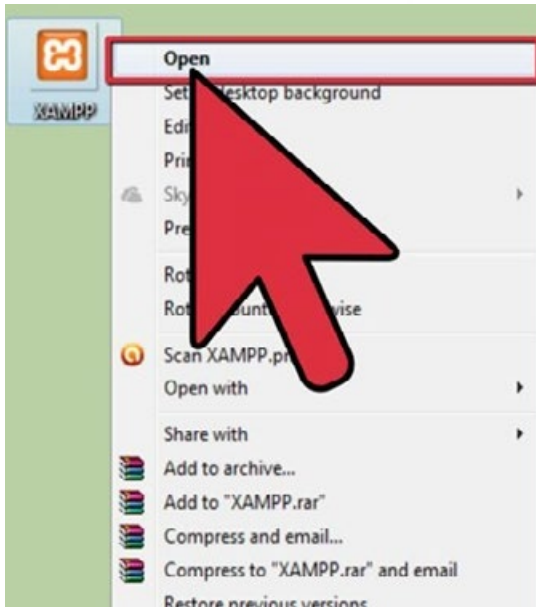


5: Acepta la configuración predeterminada. Un comando se abrirá y te ofrecerá una instalación inicial. Simplemente presiona Enter y acepta la configuración predeterminada. Para hacer más fácil la instalación, simplemente pulsa Enter cada vez que se te indique en la línea de comandos.

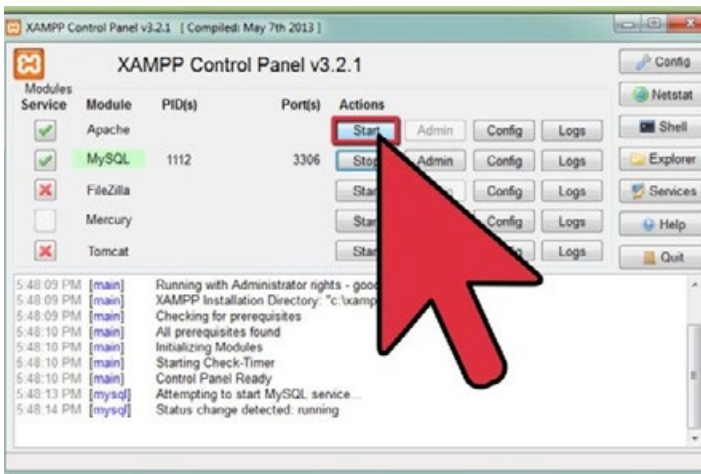
La configuración puede ser cambiada en cualquier momento en la edición de los archivos de configuración.



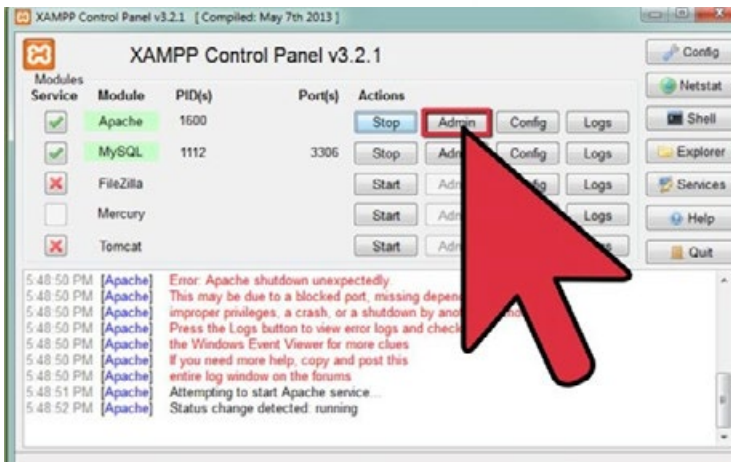
6: Cuando la instalación se haya completado, cierra la línea de comandos.



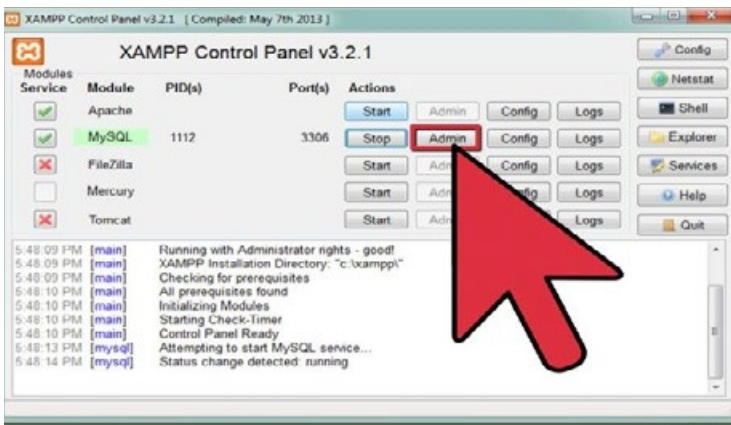
7: Inicia el panel de control de XAMPP.



8: Inicia los componentes de Apache y MySQL. También puedes iniciar los otros componentes si los vas a usar.



9: Verifica la instalación de Apache.
En el Panel de control, haz clic en el enlace administrativo de Apache.



10: Verifica la instalación de MySQL.
En el panel de control de XAMPP, haz clic en el enlace de administración MySQL.

- Si los pasos de verificación tienen éxito, XAMPP debe estar instalado correctamente en tu PC. Abre el navegador y en la barra de direcciones escribe "localhost".

GENERACIÓN Y DISEÑO DE UNA BASE DE DATOS EN MYSQL

USO DEL SHELL, COMANDOS BÁSICOS Y TABLAS:

Podemos enviar peticiones directamente al shell de MySQL, editar estas peticiones en un editor de texto separado que definimos con la variable de entorno EDITOR, o podemos utilizar un archivo con peticiones de MySQL (un script o lote) a ser ejecutadas por el intérprete de MySQL.

CONECTÁNDOSE A LA BASE DE DATOS:

Esta guía asume que ya tienes creada una base de datos, así como un usuario con los privilegios necesarios para hacer las operaciones que se requieren en la base de datos.

Los cuatro parámetros que necesitamos para establecer una conexión a la base de datos es el host donde reside la base de datos, el nombre de usuario, la contraseña y el nombre de la base de datos que vamos a manipular.

```
mysql -h [host] -D [base de datos] -u [usuario] -p
```

Esto te pedirá la contraseña, para que no sea guardada en el historial, por ejemplo:

```
mysql -h servidor.jveweb.net -D nombre_base_de_datos -u juan -p
```

Puedes especificar la contraseña en el comando agregando la contraseña junto a -p, no dejes un espacio entre -p y la contraseña para conectarte de esta manera, aunque no usar la contraseña en el comando es recomendable, por ejemplo:

```
mysql -h servidor.jveweb.net -D nombre_base_de_datos -u juan -punacontraseña
```

El parámetro -D para especificar la base de datos a usar desde que nos conectamos también es opcional, si no lo usas puedes ver una lista de las bases de datos disponibles usando show databases; y seleccionar la base de datos a usar con use [nombre base de datos]; en la línea de comandos de mysql, por ejemplo: use usuarios;

Si funcionó, obtendremos un resultado similar a este:

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6324623
Server version: 5.1.39-log MySQL Server
```

```
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.
```

```
mysql>
```

Para terminar la sesión escribe quit. Si te estás conectando a una base de datos ubicada en un host externo, es recomendable el uso de SSL al conectarse a la base de datos, para hacer esto usa el parámetro `-ssl`

ENVIANDO PETICIONES AL SHELL DE MYSQL:

Una vez que estamos en el shell de MySQL, podemos enviar peticiones de MySQL.

Para ejecutarlas tenemos que terminarlas con un punto y coma (;), o con \g, por ejemplo:

```
show tables;
```

La petición no es ejecutada hasta que el punto y coma es encontrado, esto nos permite escribir peticiones de MySQL en líneas múltiples, por ejemplo:

```
show
```

```
tables
```

```
;
```

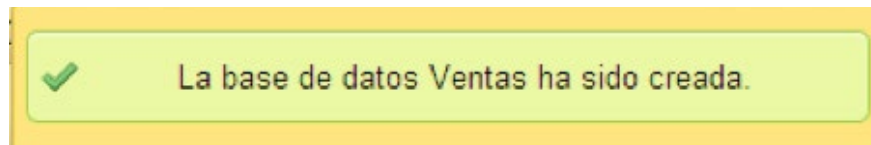
Si queremos presentar los resultados verticalmente, necesitamos terminar las peticiones con \G en vez de un punto y coma o \g

RELACIONAR TABLAS Y ESTRUCTURAR DATOS:

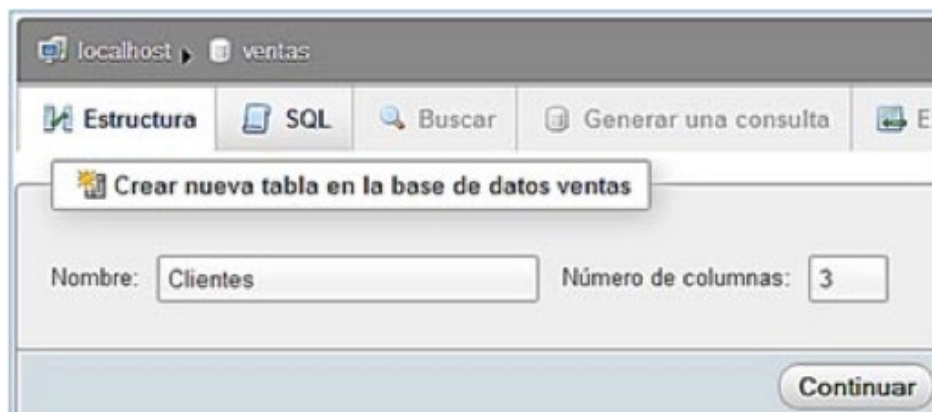
Empecemos a entrar a PHPMYADMIN y creamos una nueva base de datos en la pestaña Base de datos como muestra en la figura:



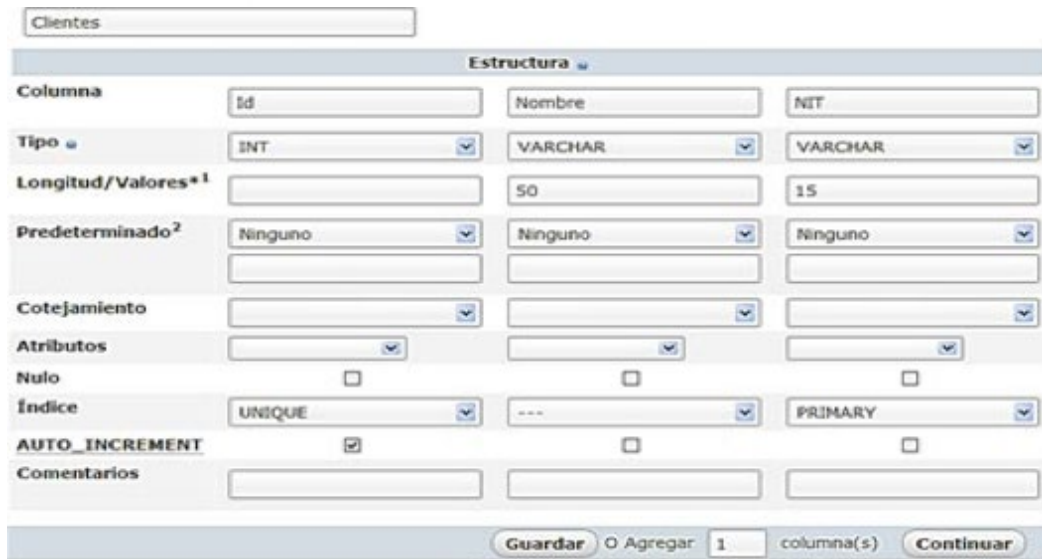
Hacemos clic en el botón CREAR y nos saldrá el aviso que se creó la base de datos:



Ahora entramos en la base de datos que creamos que es venta y creamos la primera tabla que será CLIENTES de la siguiente forma:



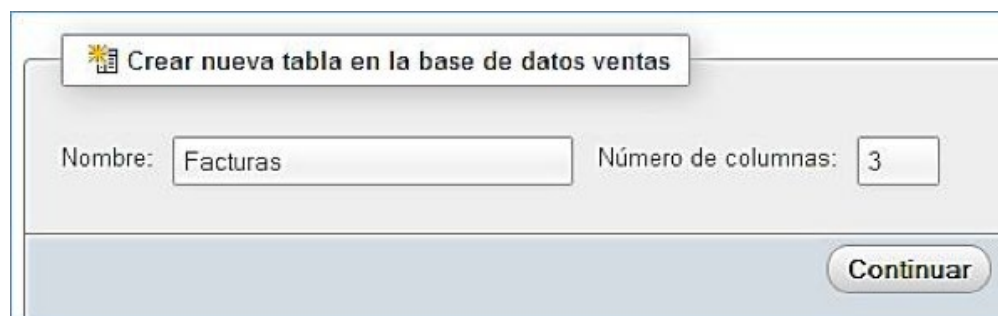
Después de hacer clic en CONTINUAR creamos los campos de la siguiente manera:




Estructura			
Columna	Id	Nombre	NIT
Tipo	INT	VARCHAR	VARCHAR
Longitud/Valores*		50	15
Predeterminado ²	Ninguno	Ninguno	Ninguno
Cotejamiento			
Atributos			
Nulo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Índice	UNIQUE	---	PRIMARY
AUTO_INCREMENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Comentarios			

Guardar O Agregar 1 columna(s) Continuar

Tomemos en cuenta que solo tenemos tres campos el ID que es la llave UNIQUE y se autoincrementa después el NOMBRE que es de tipo VARCHAR y el NIT que es tipo también VARCHAR y además que es la llave PRIMARIA luego hacemos clic en GUARDAR no en continuar si no que en GUARDAR. Y nos saldrá un aviso que dice que la tabla CLIENTES SE CREO, ahora de la misma forma también en la base de datos VENTAS creamos la tabla FACTURAS con tres campos:

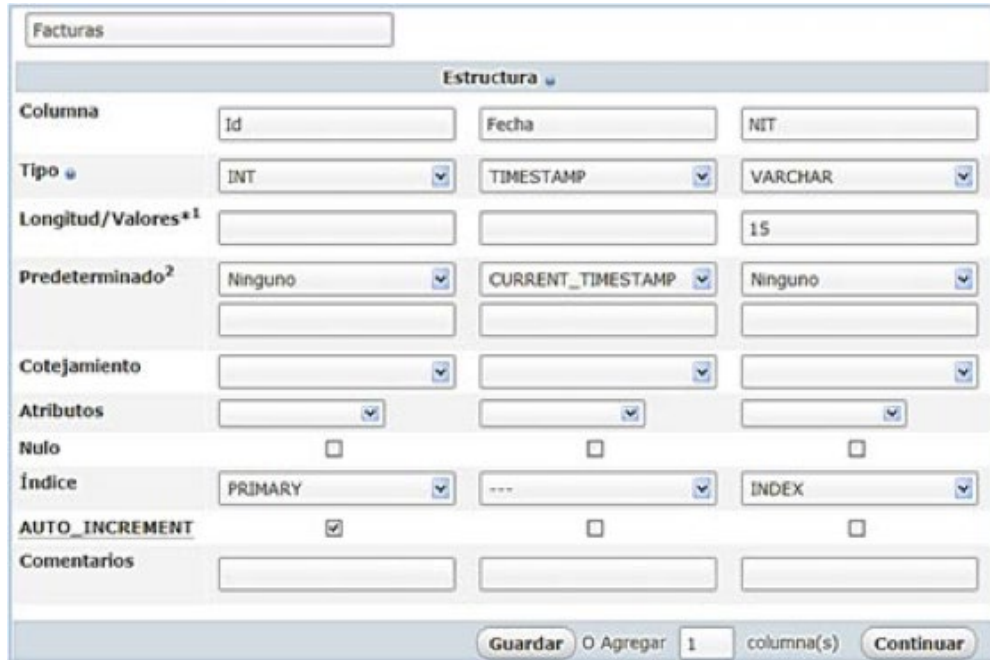


 Crear nueva tabla en la base de datos ventas

Nombre: Facturas Número de columnas: 3

Continuar

Y creamos los tres campos de la siguiente forma:



Facturas			
Estructura			
Columna	Id	Fecha	NIT
Tipo	INT	TIMESTAMP	VARCHAR
Longitud/Valores ^{*1}			15
Predeterminado ²	Ninguno	CURRENT_TIMESTAMP	Ninguno
Cotejamiento			
Atributos			
Nulo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Índice	PRIMARY	---	INDEX
AUTO_INCREMENT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Comentarios			

Guardar O Agregar 1 columna(s) Continuar

Tomemos en cuenta que se crearon tres campos el ID de tipo INT que es índice principal y se autoincrementa, luego el campo fecha de tipo TIMESTAMP como predeterminado tiene CURENT_TIMESTAMP para que obtenga la fecha actual del ordenador o el sistema y el otro campo viene siendo el NIT, que es de tipo VARCHAR de tamaño 15 y además es índice de tipo INDEX, esta campo NIT es el que vamos a relacionar con el campo NIT del cliente que también es de tipo VARCHAR eso es muy importante.

Luego hacemos clic en GUARDAR no el continuar si hacemos clic en continuar seguiremos creando más campos por eso hacemos clic en GUARDAR. Ahora nuestra base de datos se vera de la siguiente forma:




Tabla	Acción	Filas	Tipo
<input type="checkbox"/> clientes	Examinar Estructura Buscar Insertar Vaciar Eliminar		InnoDB
<input type="checkbox"/> Facturas	Examinar Estructura Buscar Insertar Vaciar Eliminar		InnoDB
1 tabla	Número de filas		InnoDB

☐ Marcar todos / Desmarcar todos
 Para los elementos que están marcados: ▼

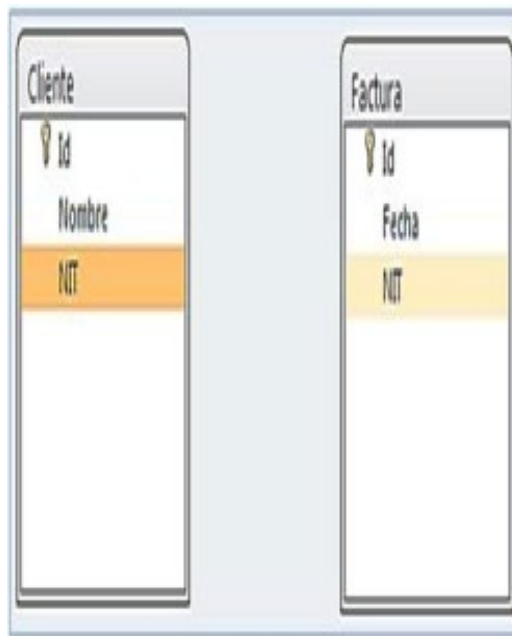
Ahora vamos a crear la siguiente relación:



Significa que un cliente puede tener muchas facturas y una factura le pertenece a un cliente solo a un cliente.

Entonces para eso tenemos que relacionar las facturas por supuesto que también podíamos a ver creado en la tabla FACTURA un campo llamado ID_CLIENTE y relacionar el ID de CLIENTES con el ID_CLIENTES de FACTURAS que también da la misma relación, pero nosotros vamos a optimizar usando el NIT en ambas tablas.

Entonces hacemos clic en la tabla FACTURAS:



Y luego en VISTA RELACIONES:

#	Columna	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
<input type="checkbox"/> 1	<u>Id</u>	int(11)			No	Ninguna
<input type="checkbox"/> 2	Fecha	timestamp			No	CURRENT_TIMEST
<input type="checkbox"/> 3	NIT	int(11)			No	Ninguna
<input type="checkbox"/> Marcar todos / Desmarcar todos Para los elementos que están marcados:						
<input type="checkbox"/> Vista de impresión <input checked="" type="checkbox"/> Vista de relaciones <input type="checkbox"/> Planteamiento de la estructura de						
<input type="checkbox"/> Agregar <input type="text" value="1"/> columna(s) <input checked="" type="radio"/> Al final de la tabla <input type="radio"/> Al comienzo de la tabla						

Luego como dijimos vamos a relacionar el NIT de FACTURAS con el NIT de CLIENTES como se muestra en la siguiente imagen:

Columna	Restricción de clave foránea (INNODB)
Id	<input type="text"/>
Fecha	¡No se ha definido ningún índice!
NIT	<input type="text" value="ventas.clientes.NIT"/> ON DELETE <input type="text" value="CASCADE"/> ON UPDATE <input type="text" value="CASCADE"/>

En NIT escogemos 'VENTAS'.CLIENTES.'NIT' en ON DELETE escogemos CASCADE y en ON UPDATE también CASCADE, para que se elimine y actualice en cascada para mantener la integridad de datos. Luego hacemos clic en GUARDAR.

Recuerda que primero antes de crear una factura tenemos que ya tener Clientes creados con sus NIT respectivos y después podrás crear Facturas.

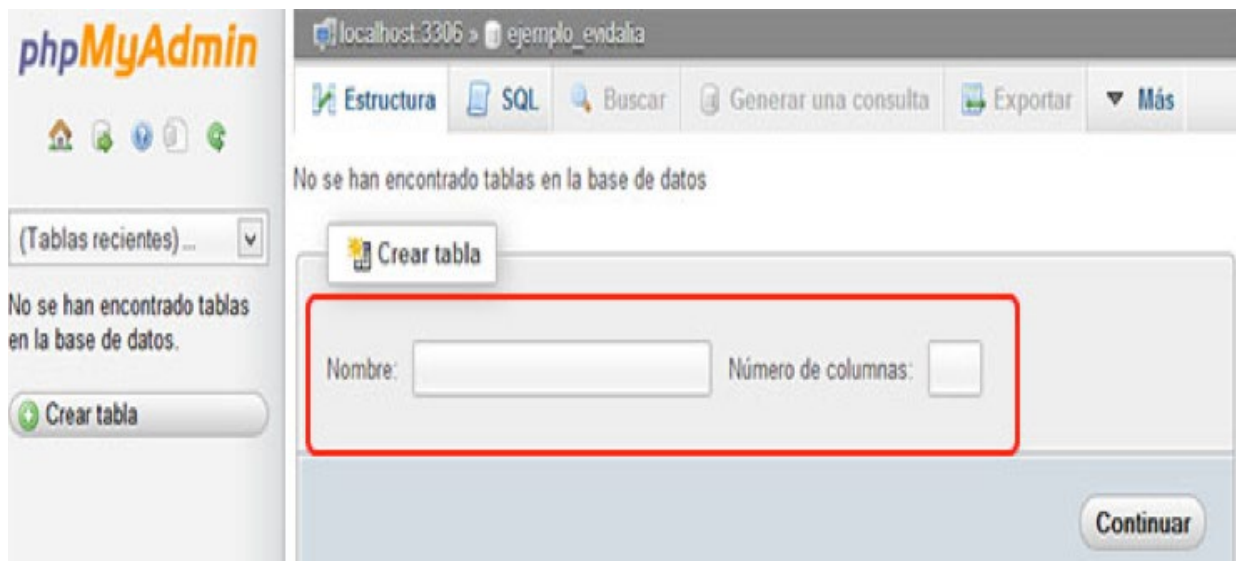
MANIPULACIÓN AVANZADA DE DATOS EN MYSQL

CONSULTAS, CONDICIONALES Y OPERACIONES MATEMÁTICAS EN MYSQL:

Creación de la primera tabla.

Una vez dentro de phpMyAdmin y en el caso de que no tengamos ninguna tabla creada, accederemos directamente a la ventana para la creación de una nueva tabla, donde deberemos indicar un nombre y el número de columnas que contendrá. Aunque deberíamos hacer un estudio de los campos necesarios en cada una de las tablas, es posible que necesitemos añadir o eliminar campos. Por ello no es excesivamente importante el número de campos de la tabla.

Para nuestro ejemplo vamos a crear la tabla clientes con 5 campos. Indicaremos el valor "clientes" como Nombre y en Número de columnas indicaremos 5.



Una vez indicados los datos, nos aparecerá una nueva ventana en la que deberemos indicar el **Nombre del campo** y su **Tipo**.

Para los nombres de campo hemos elegido: *nombre, dirección, código postal, localidad, provincia*
Para el tipo de campo podremos seleccionar entre los tipos básicos:

INT: Para valores numéricos

VARCHAR: Para campos de texto corto

TEXT: Para campos de texto largos

DATE: Para valores de fecha y hora

En nuestro caso vamos a seleccionar todos los campos como **VARCHAR**, excepto el código postal, que será numérico (**INT**).

Finalmente deberemos indicar la longitud del campo, es decir, el número de caracteres que podrá almacenar dicho campo.

Nombre de la tabla: Agregar columna(s)

Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento	Atributos
<input type="text" value="nombre"/>	<input type="text" value="VARCHAR"/>	<input type="text" value="150"/>	<input type="text" value="Ninguno"/>	<input type="text" value=""/>	<input type="text" value=""/>
<input type="text" value="direccion"/>	<input type="text" value="VARCHAR"/>	<input type="text" value="300"/>	<input type="text" value="Ninguno"/>	<input type="text" value=""/>	<input type="text" value=""/>
<input type="text" value="cod_postal"/>	<input type="text" value="INT"/>	<input type="text" value="8"/>	<input type="text" value="Ninguno"/>	<input type="text" value=""/>	<input type="text" value=""/>
<input type="text" value="localidad"/>	<input type="text" value="VARCHAR"/>	<input type="text" value="200"/>	<input type="text" value="Ninguno"/>	<input type="text" value=""/>	<input type="text" value=""/>
<input type="text" value="provincia"/>	<input type="text" value="VARCHAR"/>	<input type="text" value="200"/>	<input type="text" value="Ninguno"/>	<input type="text" value=""/>	<input type="text" value=""/>

Comentarios de la tabla:

Motor de almacenamiento:

Cotejamiento:

definición de la PARTICIÓN:

Finalmente, pulsaremos sobre el botón Guardar, para almacenar la estructura de la tabla.

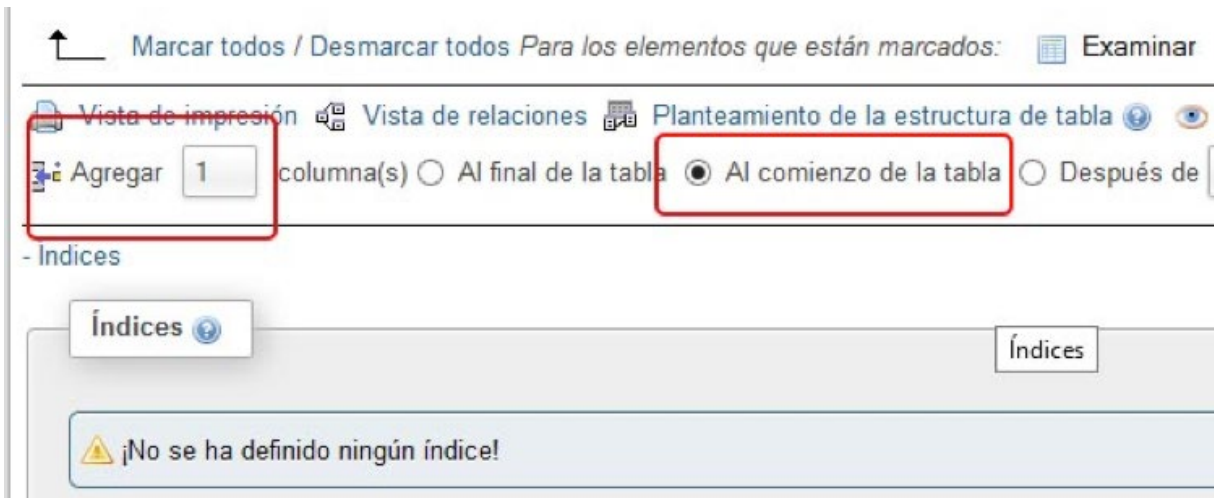
Vemos que en la zona izquierda nos ha aparecido el icono de la nueva tabla. Pulsaremos sobre él y nos aparecerá la estructura de la tabla (ya que no existen registros para mostrar).

AÑADIENDO UN ÍNDICE A LA TABLA:

Un poco más abajo de la estructura de la tabla, nos aparece el enlace índices. Si pulsamos sobre él se desplegará dicha sección, mostrándonos la advertencia "No se ha definido ningún índice".

En el caso de tener alguna referencia o algún código numérico único para cada valor, podríamos hacer que dicha columna fuera un índice, pero en nuestro caso no existe ninguno, así que vamos a crear una nueva columna que servirá de índice.

Debajo de la estructura de la tabla, indicaremos en **Agregar** el valor 1, marcaremos **Al comienzo de la tabla** y pulsaremos **Continuar**.



Marcar todos / Desmarcar todos Para los elementos que están marcados: Examinar

Vista de impresión Vista de relaciones Planteamiento de la estructura de tabla

Agregar 1 columna(s) ☐ Al final de la tabla ☒ Al comienzo de la tabla ☐ Después de

- Índices

Índices

¡No se ha definido ningún índice!

En la nueva ventana que nos aparecerá indicaremos como **Nombre:** id, **Tipo:** INT, **Índice:** Primary, activaremos la casilla de **Auto_Increment** y pulsaremos sobre el botón **Guardar**.



Agregar columnas

Estructura

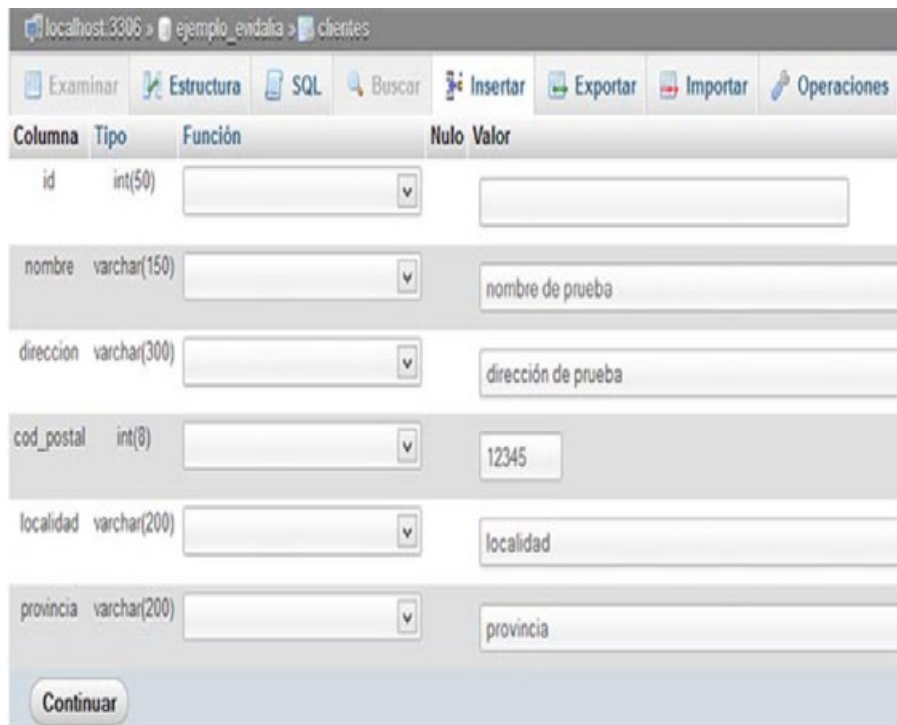
Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento	Atributos	Nulo	Índice	A_I	Comentarios	MIME
id	INT	50	Ninguno			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>		

Guardar

AÑADIENDO REGISTROS A LA TABLA:

Una vez tenemos nuestra estructura correctamente creada, es el momento de agregar la información a la tabla y desde el propio phpMyAdmin, podremos realizar esta operación.

Una vez seleccionada nuestra tabla Clientes, pulsaremos sobre la pestaña **Insertar**, situada en la zona superior. Vamos a añadir una única fila, así que en la nueva ventana que nos aparece, indicaremos los valores y pulsaremos en **Continuar**.

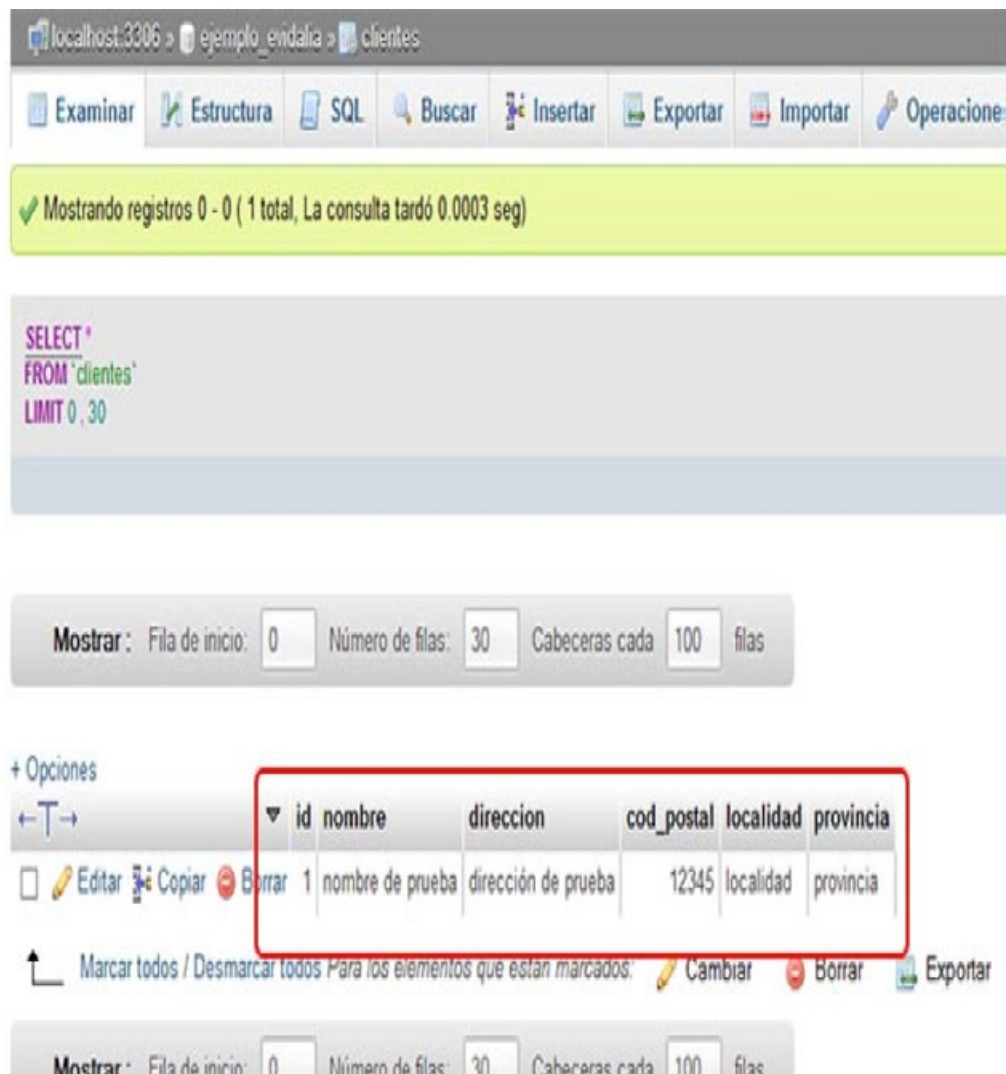


Columna	Tipo	Función	Nulo	Valor
id	int(50)			
nombre	varchar(150)			nombre de prueba
direccion	varchar(300)			dirección de prueba
cod_postal	int(8)			12345
localidad	varchar(200)			localidad
provincia	varchar(200)			provincia

Continuar

No hemos indicado nada en el campo id, ya que será la propia base de datos la encargada de gestionar y numerar correctamente a cada registro insertado.

Finalmente pulsaremos sobre la pestaña superior **Examinar**, para ver como efectivamente se ha creado una fila con los valores que hemos indicado.



localhost:3306 » ejemplo_endalla » clientes

Examinar Estructura SQL Buscar Insertar Exportar Importar Operaciones

✓ Mostrando registros 0 - 0 (1 total, La consulta tardó 0.0003 seg)

```
SELECT *
FROM 'clientes'
LIMIT 0,30
```

Mostrar: Fila de inicio: 0 Número de filas: 30 Cabeceras cada 100 filas

+ Opciones

	id	nombre	direccion	cod_postal	localidad	provincia
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	nombre de prueba	dirección de prueba	12345	localidad	provincia

Marcar todos / Desmarcar todos Para los elementos que están marcados: Cambiar Borrar Exportar

Mostrar: Fila de inicio: 0 Número de filas: 30 Cabeceras cada 100 filas

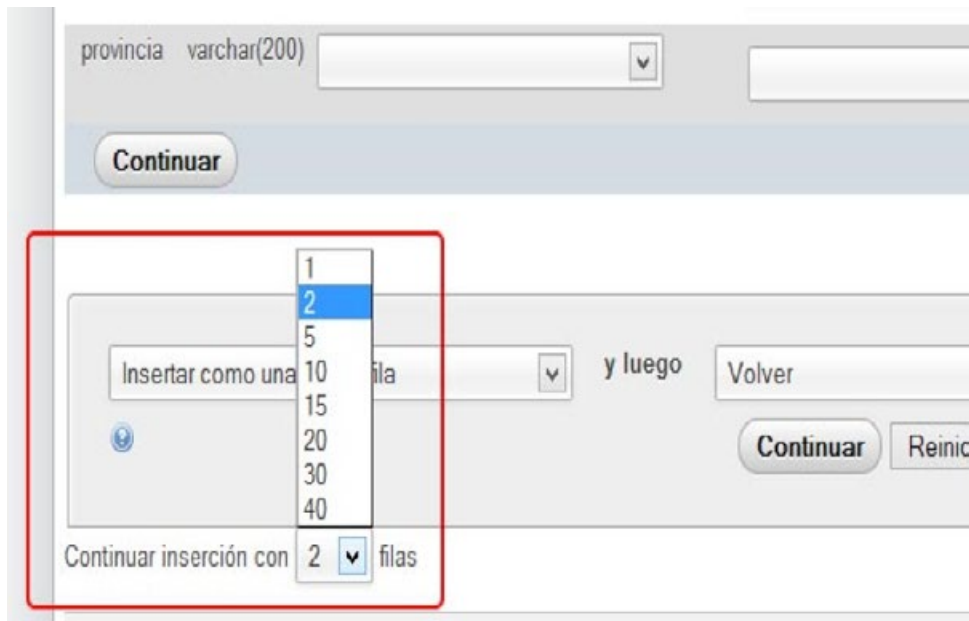
No hemos indicado nada en el campo id, ya que será la propia base de datos la encargada de gestionar y numerar correctamente a cada registro insertado.

Finalmente pulsaremos sobre la pestaña superior **Examinar**, para ver como efectivamente se ha creado una fila con los valores que hemos indicado.

AÑADIR REGISTROS:

Podemos insertar registros pulsando sobre la pestaña **Insertar** situada en la zona superior.

Podemos insertar una única línea rellenando los datos y pulsando sobre el botón **Continuar** de dicha sección, o bien, podemos indicar un número concreto de líneas y rellenar todos los campos de una única vez. Para indicar el número de registros a insertar nos situaremos en la zona inferior de phpMyAdmin y elegiremos la cantidad de registros desde el desplegable **Continuar inserción con ... filas**



The screenshot shows the 'Insertar' (Insert) tab in phpMyAdmin. At the top, there is a form for inserting a single record, with a 'provincia' field (varchar(200)) and a 'Continuar' button. Below this, there is a section for inserting multiple records. A red box highlights the 'Continuar inserción con ... filas' dropdown menu, which is currently set to '2'. The dropdown menu is open, showing a list of options: 1, 2, 5, 10, 15, 20, 30, and 40. The '2' option is selected. To the right of the dropdown, there is a 'y luego' (and then) dropdown menu and a 'Volver' (Back) button. At the bottom right, there are 'Continuar' and 'Reiniciar' (Reset) buttons.

Aparecerá una ventana con la cantidad de formularios que acabamos de indicar, es entonces cuando indicaremos los datos a insertar. Esta opción es muy útil cuando tengamos valores repetitivos y necesitemos copiar y pegar los valores de los campos.

Hay que tener en cuenta que existe en cada formulario una casilla llamada **Ignorar**.



The screenshot shows a web interface for inserting data into a MySQL database. At the top, there is a button labeled "Continuar". Below it, a checkbox labeled "Ignorar" is checked and highlighted with a red rectangle. Below the checkbox is a table with the following structure:

Columna	Tipo	Función	Nulo	Valor
id	int(50)	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>
nombre	varchar(150)	<input type="text"/>	<input type="checkbox"/>	<input type="text"/>

Cuando el formulario contenga valores, la casilla se desactiva automáticamente, pero en el caso de que el formulario esté vacío, Ignorar permanecerá activado. Esto nos servirá para cuando hayamos creado más formularios de los necesarios. Así phpMyAdmin solo insertará los campos que contengan algún valor, evitando la inserción de registros vacíos.



The screenshot shows the bottom part of the MySQL data entry form. It features a dropdown menu labeled "Insertar como una nueva fila" with a downward arrow. To its right, the text "y luego" is followed by another dropdown menu labeled "Volver" with a downward arrow, which is highlighted with a red rectangle. Below these are two buttons: "Continuar" and "Reiniciar". At the bottom, there is a label "continuar inserción con" followed by a dropdown menu showing the number "2" and the word "filas".

Finalmente, podemos indicar la acción que se realizará después de pulsar el botón de **Continuar** (para la inserción de los datos). Para ello tenemos el desplegable y luego en el que podremos seleccionar **Volver** para regresar al listado de registros o bien **Insertar un nuevo registro**, para permanecer en la página actual y seguir insertando valores.

MODIFICAR LOS REGISTROS:

Una vez insertados los registros, nos hemos dado cuenta de que tenemos un error en algún valor.

Necesitamos realizar la modificación de dicho valor. Para ello phpMyAdmin nos ofrece las herramientas necesarias.

Lo primero será acceder a la pestaña de visualización de los registros y para ello, pulsaremos sobre la pestaña **Examinar**.

Nos aparece nuestra tabla con los registros que hemos ido insertado y vemos que por cada fila o registro, disponemos de unos botones **Editar**.



Pulsando el botón editar de una fila, podremos modificar su contenido y posteriormente guardar los cambios.

Si queremos copiar una fila, pulsaremos sobre **Copiar** y accederemos al formulario donde si queremos, podremos modificar los valores necesarios.

Después y como siempre, pulsaremos **Continuar** para almacenar los valores.

ELIMINAR LOS REGISTROS:

Como es lógico, en el caso de querer eliminar alguna fila, pulsaremos sobre **Borrar** de dicha fila.

Deberemos tener en cuenta dos aspectos fundamentales:










- En el caso de eliminar un registro, perderemos dicha información y no podremos recuperarla, con lo que deberemos tener especial cuidado.
- El valor de índice es autonumérico y cuando se genera un valor, aunque se elimine, al crear un nuevo registro, su índice continuará con el siguiente número correlativo.




OPERACIONES EN MASA:

También tenemos la posibilidad de eliminar o modificar valores en masa, es decir, varios o todos a la vez. Para ello, deberemos marcar las casillas de las filas que queramos seleccionar, o bien pulsar sobre **Marcar todos** para seleccionarlas todas.

A continuación disponemos de los botones **Cambiar** y **Eliminar** para actuar sobre todos los registros seleccionados.

+ Opciones

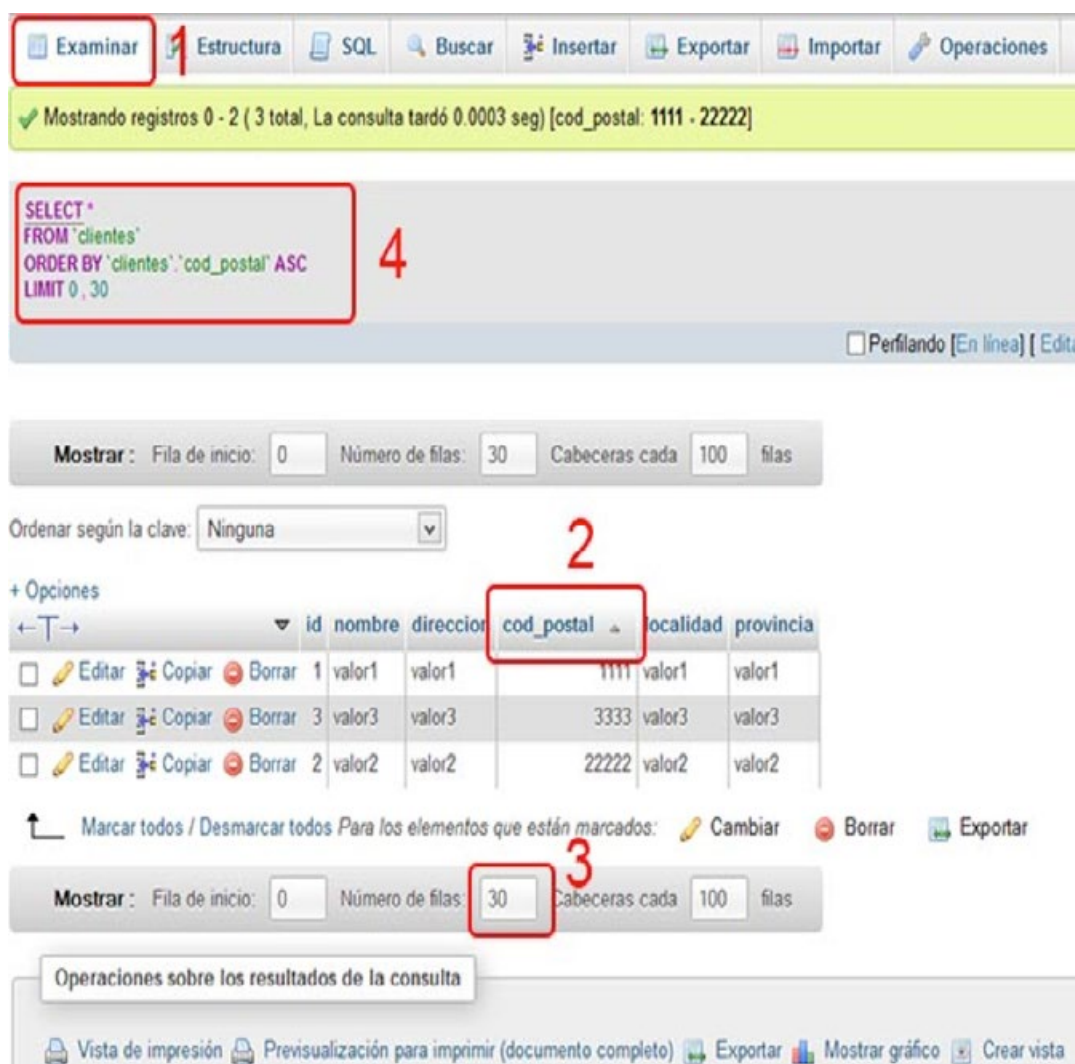
					id	nombre	direccion	cod_postal	localidad	provincia		
<input checked="" type="checkbox"/>		Editar		Copiar		Borrar	1	valor1	valor1	1111	valor1	valor1
<input checked="" type="checkbox"/>		Editar		Copiar		Borrar	2	valor2	valor2	22222	valor2	valor2
<input checked="" type="checkbox"/>		Editar		Copiar		Borrar	3	valor3	valor3	3333	valor3	valor3

 **Marcar todos / Desmarcar todos** Para los elementos que están marcados:  **Cambiar**  **Borrar**

Mostrar : Fila de inicio: 0 Número de filas: 30 Cabeceras cada 100 filas

SELECCIONAR TODOS LOS REGISTROS DE UNA TABLA:

Esta es la opción más sencilla. Si queremos mostrar todos los registros de una tabla, tan solo deberemos pulsar en la pestaña **Examinar** (1) situada en la zona superior. Al pulsar dicha pestaña, veremos todo el contenido de nuestra tabla ordenados según los hemos insertado



1 Examinar Estructura SQL Buscar Insertar Exportar Importar Operaciones

Mostrando registros 0 - 2 (3 total, La consulta tardó 0.0003 seg) [cod_postal: 1111 - 22222]

```
SELECT *
FROM 'clientes'
ORDER BY 'clientes'.cod_postal ASC
LIMIT 0, 30
```

4

Mostrar: Fila de inicio: 0 Número de filas: 30 Cabeceras cada 100 filas

Ordenar según la clave: Ninguna

2

+ Opciones

	id	nombre	direccion	cod_postal	localidad	provincia
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	valor1	valor1	1111	valor1	valor1
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	3	valor3	valor3	3333	valor3	valor3
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	2	valor2	valor2	22222	valor2	valor2

3

Mostrar: Fila de inicio: 0 Número de filas: 30 Cabeceras cada 100 filas

Operaciones sobre los resultados de la consulta

Vista de impresión Previsualización para imprimir (documento completo) Exportar Mostrar gráfico Crear vista

Podremos personalizar dicha visualización ordenando los registros según nos convenga. Tan solo deberemos pulsar sobre el **título de la columna** (2) para que los registros se ordenen de forma ascendente o descendente.

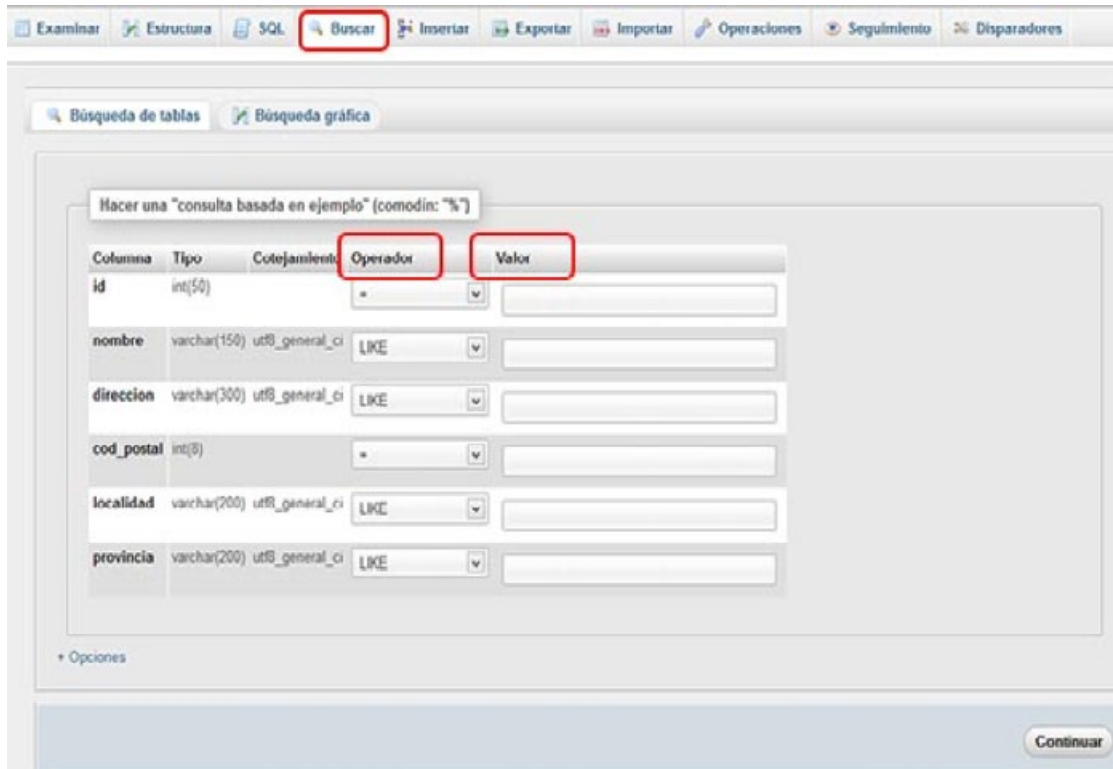
También podremos indicar el número de registros a mostrar, es decir, realizar una paginación de los resultados. Para ello indicaremos el **número de filas** (3) y pulsaremos sobre mostrar.

Cada cambio que realizamos en la visualización de los registros, se refleja en la **zona del código SQL** (4). Esto nos puede ser útil para cuando queramos indicar nuestra propia consulta SQL.

Selección de registros mediante la opción Buscar

Aparte de mostrar todos los registros de una tabla, podemos utilizar la herramienta **Buscar** para hacer un filtrado de los registros a mostrar.

Para utilizar dicha herramienta, pulsaremos sobre la pestaña **Buscar** y nos aparecerán todos los campos de nuestra tabla, un **operador** y a continuación un campo de texto donde podremos indicar un **valor**:



Columna	Tipo	Cotejamiento	Operador	Valor
id	int(50)		=	
nombre	varchar(150) utf8_general_ci		LIKE	
direccion	varchar(300) utf8_general_ci		LIKE	
cod_postal	int(5)		=	
localidad	varchar(200) utf8_general_ci		LIKE	
provincia	varchar(200) utf8_general_ci		LIKE	

Los operadores de filtrado mas comunes para texto serian:

LIKE: Para seleccionar campos que empiecen como el valor.

LIKE %...%: Para seleccionar campos que contengan la cadena indicada.

NOT LIKE: Para seleccionar los campos que no contengan la cadena.

= Para seleccionar campos cuyo texto sea igual.

=" Para seleccionar registros cuyo valor sea nulo.

Y en el caso de los campos numéricos:

=” Para seleccionar registros cuyo valor sea nulo.

= Para mostrar los registros cuyo valor sea igual al indicado.

> Para mostrar los registros cuyo valor sea mayor al indicado.

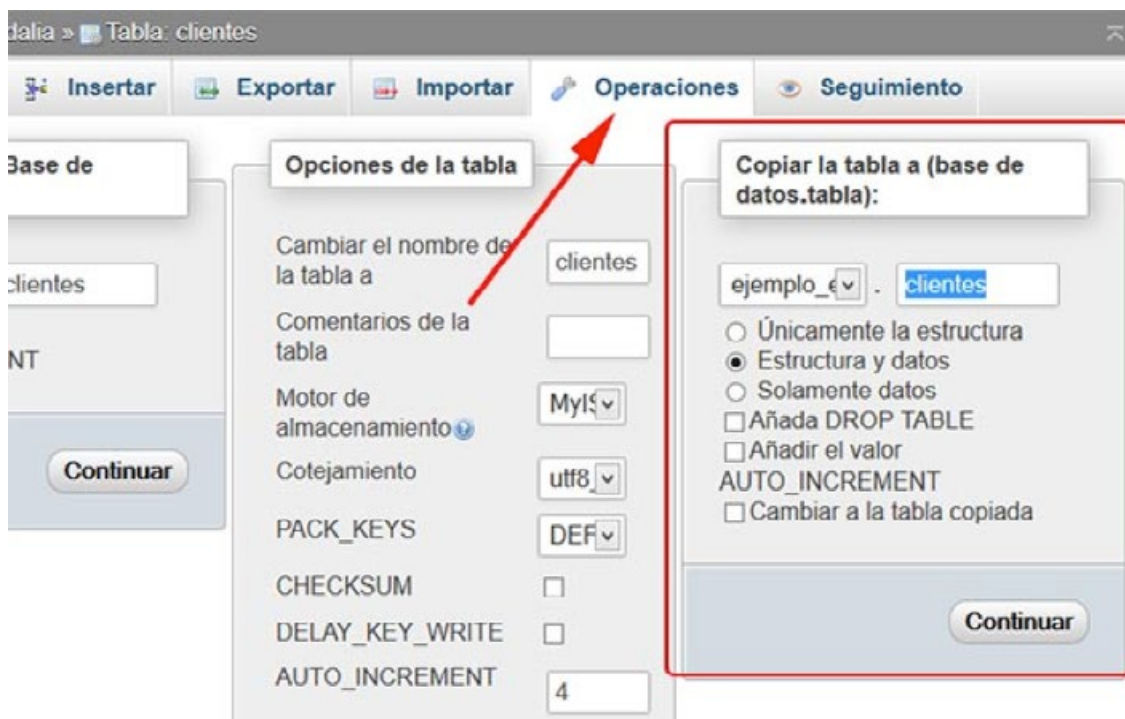
>= Para mostrar los registros cuyo valor sea igual o mayor al indicado.

VALIDACIÓN Y REDUNDANCIA DE DATOS:

Copiar tablas

Anteriormente habíamos creado una tabla llamada “clientes”. Vamos a suponer que queremos tener en nuestra base de datos otra tabla llamada “proveedores” y que va a tener la misma estructura.

Lo más sencillo y rápido será crear una copia de la tabla “clientes”. Para ello, seleccionaremos dicha tabla y pulsaremos sobre la pestaña **Operaciones** situada en la zona superior de phpMyAdmin. Deberemos localizar el panel **Copiar la tabla a (base de datos.tabla)**



Desde este panel podremos seleccionar la base de datos de destino y cambiar el nombre de la tabla, que en nuestro ejemplo será proveedores.

En cuanto a las opciones, deberemos seleccionar una de estas tres:

- **Únicamente la estructura:** Para hacer una copia de la tabla pero sin ningún registro.
- **Estructura y datos:** Para hacer una copia exacta de la tabla
- **Solamente datos:** Para copiar únicamente los registros.

Hay que tener en cuenta que la tabla de destino ya debe estar creada y debe poseer concordancia en los campos ya que de no ser así, se producirían errores o incluso la imposibilidad de realizar la copia.

Los valores para nuestro ejemplo serán, la misma base de datos, el nombre de la tabla “proveedores” y la opción “estructura y datos”.

Vaciar y borrar tablas

Hemos copiado expresamente los registros de la tabla, para mostraros a continuación como se puede eliminar su contenido.

Una forma sería pulsando la pestaña **Examinar**, seleccionando **Marcar todos** y pulsando sobre **Borrar**.



Esta forma tiene el inconveniente de que si hay muchos valores, debemos realizar el borrado página a página. La opción que mostramos, nos permitirá con un simple clic de ratón eliminar todos los valores.

De nuevo deberemos acceder a **Operaciones**, teniendo seleccionada la tabla cuyos registros queremos eliminar. Deberemos buscar el grupo **Borrar datos o tabla**. Una vez localizado, pulsaremos en el enlace **Vaciar la tabla (TRUNCATE)**.

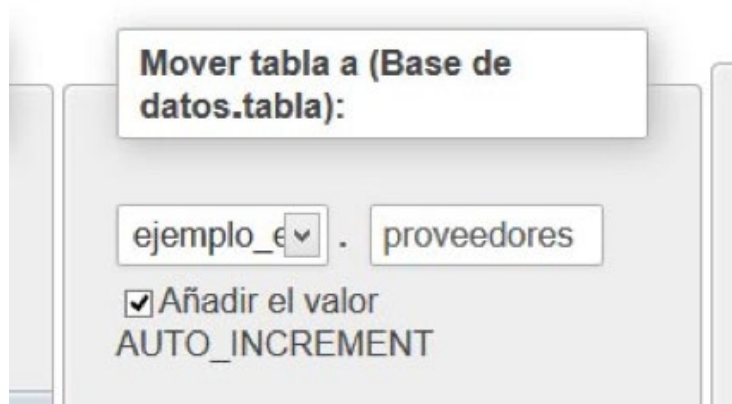


El sistema nos solicitará la confirmación para llevar a cabo la acción y tras breves instantes podremos comprobar como en nuestra tabla proveedores ya no existe ningún registro.

De igual forma, si lo que queremos es eliminar la tabla, pulsaremos sobre el enlace **Borrar la tabla (DROP)**, respondiendo de forma afirmativa a la pregunta de seguridad.

MOVER Y RENOMBRAR TABLAS:

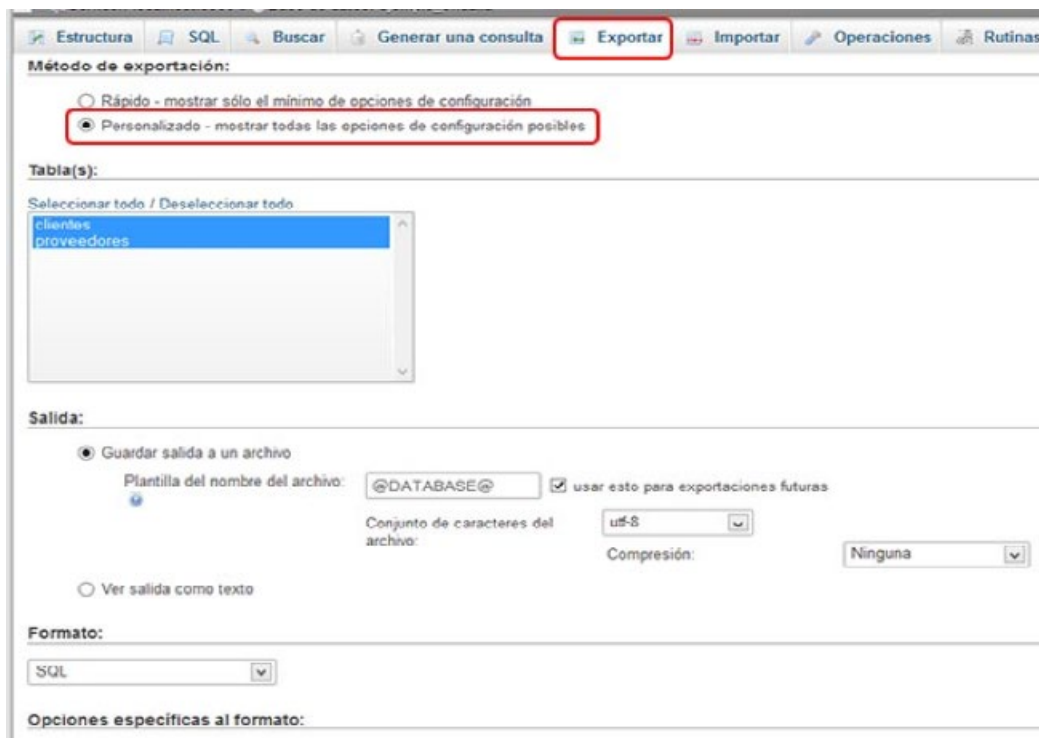
El último caso que vamos a tratar hoy será el de mover y renombrar tablas. De nuevo desde la pestaña Operaciones, encontraremos una sección llamada **Mover tabla a (Base de datos.tabla)**



Desde aquí podremos cambiar el nombre a una tabla indicando su nuevo nombre y pulsando **Continuar**, o bien en el caso de tener varias bases de datos en nuestro servidor, podremos trasladar nuestra tabla, de una base de datos a otra.

CREACIÓN DE LA COPIA DE SEGURIDAD:

Para crear el backup de nuestra base de datos desde phpMyAdmin, seleccionaremos una base de datos y a continuación deberemos acceder a la pestaña **Exportar**. En esta ventana, deberemos elegir el método de exportación. Podemos seleccionar **Rápido** si nos parecen acertadas las opciones por defecto. En cambio, pulsaremos **Personalizado** si queremos modificar alguna de estas opciones.



The screenshot shows the 'Exportar' (Export) tab in phpMyAdmin. The 'Método de exportación:' (Export method) section has two radio buttons: 'Rápido - mostrar sólo el mínimo de opciones de configuración' (selected) and 'Personalizado - mostrar todas las opciones de configuración posibles'. Below this, the 'Tabla(s):' (Table(s)) section shows a list of tables: 'clientes' and 'proveedores'. The 'Salida:' (Output) section has two radio buttons: 'Guardar salida a un archivo' (selected) and 'Ver salida como texto'. Under 'Guardar salida a un archivo', there are fields for 'Plantilla del nombre del archivo:' (containing '@DATABASE@'), 'Conjunto de caracteres del archivo:' (set to 'utf-8'), and 'Compresión:' (set to 'Ninguna'). There is also a checkbox 'usar esto para exportaciones futuras'. The 'Formato:' (Format) section has a dropdown menu set to 'SQL'. The 'Opciones específicas al formato:' (Format-specific options) section is visible at the bottom.

Para nuestro ejemplo seleccionaremos la opción Personalizado y vemos las siguientes opciones:

- **Tablas:** Para poder seleccionar las tablas que deseamos exportar.
- **Salida:** Donde podremos indicar el nombre del fichero a descargar, el conjunto de caracteres y el tipo de compresión, opción necesaria en el caso de tener un gran número de registros.
- **Formato:** Por lo general será **SQL** para restaurar más fácilmente, aunque podemos elegir otro formato si queremos visualizar los registros en otra aplicación. Por norma general para intercambiar información entre bases de datos se utiliza el formato **CSV**.

- **Opciones específicas al formato:** Aquí podemos indicar si queremos exportar la estructura, los datos o ambos.

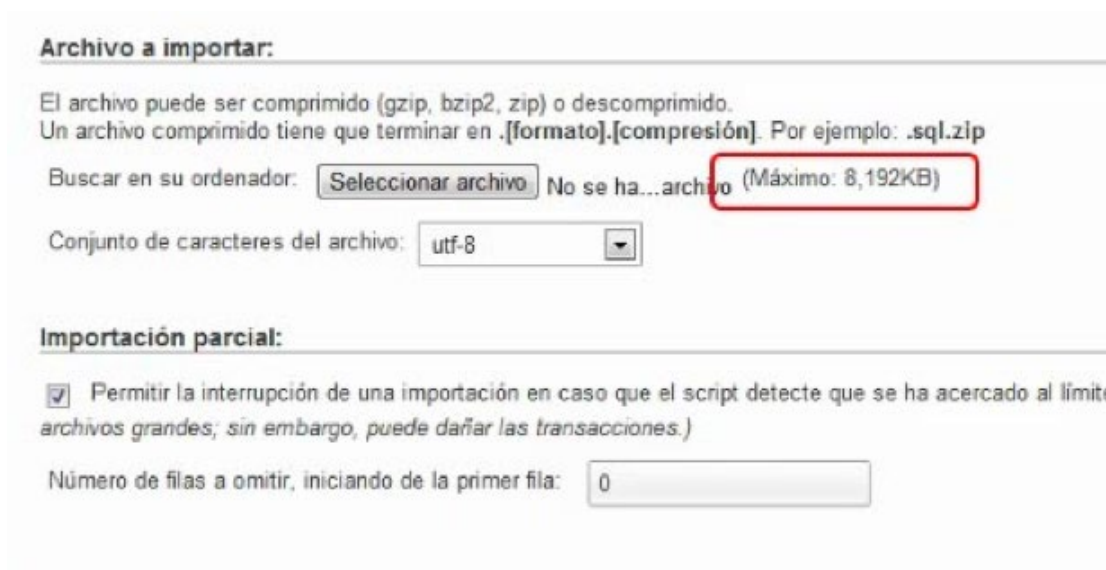
El resto de opciones las dejaremos por defecto.

Una vez pulsado el botón **Continuar**, podremos descargar el fichero generado. Dependiendo de la cantidad de información almacenada en nuestra base de datos, obtendremos un fichero de unos pocos kilobytes o de hasta varios megas o teras.

RESTAURAR LA COPIA DE SEGURIDAD:

Para realizar la opción inversa, es decir, la importación de la copia de seguridad a la base de datos, deberemos pulsar sobre la pestaña **Importar**.

Aquí tenemos muchas menos opciones que en el caso anterior. Lo que debemos tener en cuenta es el tamaño máximo del archivo a subir, ya que el sistema no nos permitirá restaurar copias de seguridad que superen dicho tamaño.



Archivo a importar:

El archivo puede ser comprimido (gzip, bzip2, zip) o descomprimido.
Un archivo comprimido tiene que terminar en **[formato].[compresión]**. Por ejemplo: **.sql.zip**

Buscar en su ordenador: No se ha seleccionado archivo (Máximo: 8,192KB)

Conjunto de caracteres del archivo:

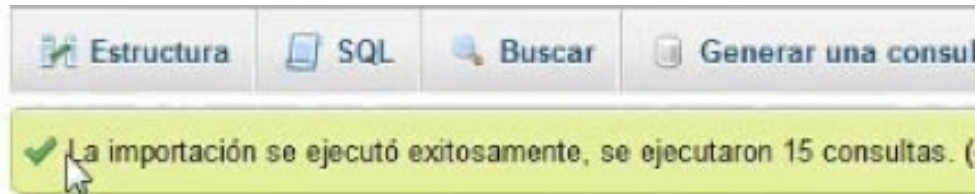
Importación parcial:

☒ Permitir la interrupción de una importación en caso que el script detecte que se ha acercado al límite archivos grandes; sin embargo, puede dañar las transacciones.)

Número de filas a omitir, iniciando de la primer fila:

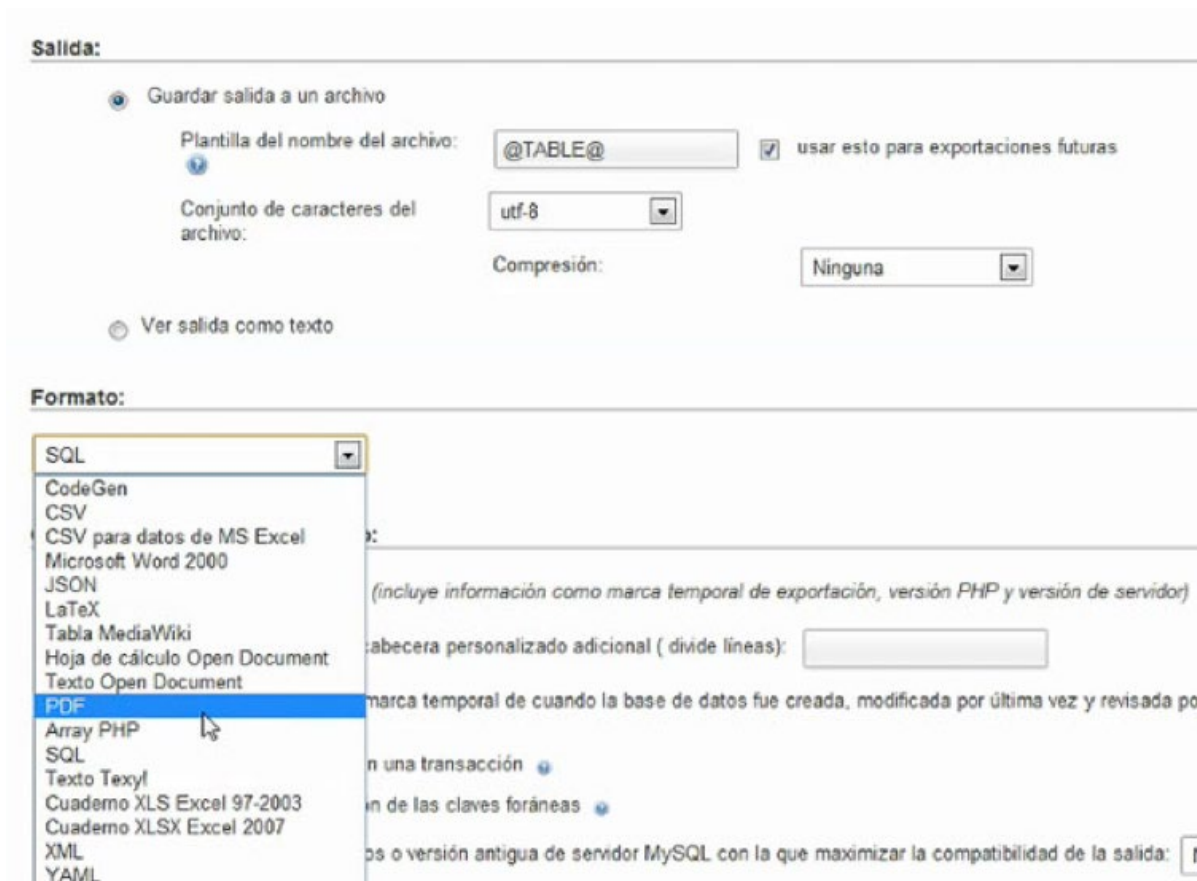
Dentro de la sección **Archivo a importar** elegiremos el fichero a subir y también elegiremos el tipo de codificación. En Importación parcial es interesante que la casilla **Permitir la interrupción...** esté activada en el caso de que nuestro fichero tenga un tamaño considerable y preveemos que la importación va a necesitar de bastante tiempo.

Por lo general, la opción **Formato** será automodificada en función del fichero que hayamos indicado. A continuación pulsaremos el botón **Continuar** y tras unos instantes tendremos nuestra copia de seguridad volcada a la base de datos. Tanto si la transacción a sido correcta como si han habido errores, veremos un mensaje informándonos de ello.



EXPORTAR TABLAS EN PDF:

Puede darse el caso de que necesitemos cierta información o bien en PDF o simplemente impresa en papel. Desde el propio entorno de phpMyAdmin, podremos crear fácilmente fichero PDF el cual podremos almacenar, enviar por correo electrónico o simplemente imprimir.

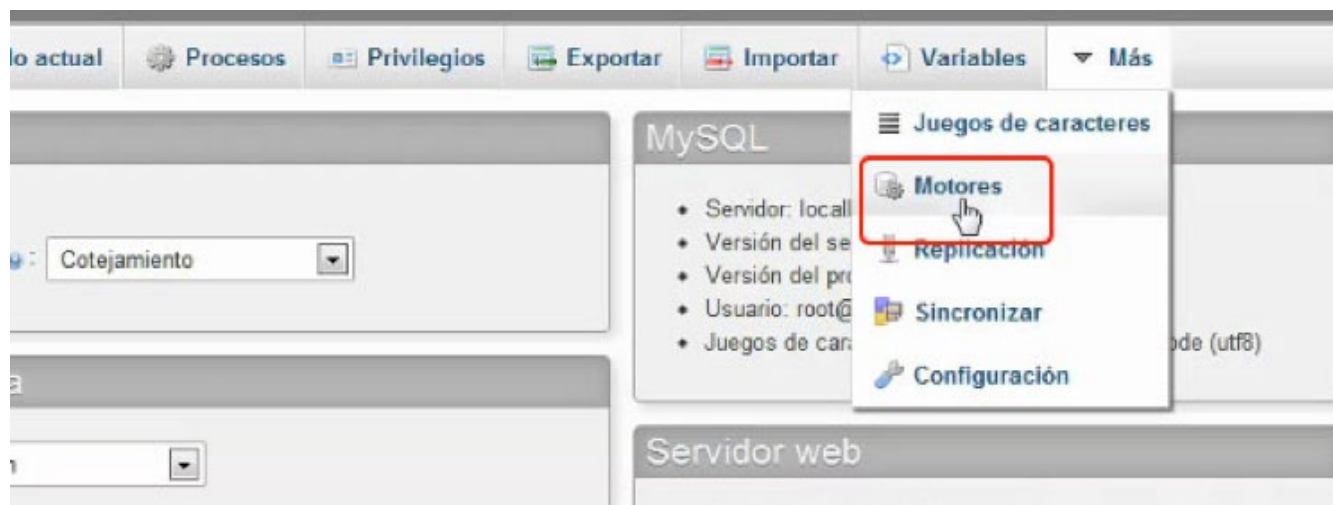


The screenshot shows the phpMyAdmin export interface. The 'Salida:' section has two radio buttons: 'Guardar salida a un archivo' (selected) and 'Ver salida como texto'. Under 'Guardar salida a un archivo', there are fields for 'Plantilla del nombre del archivo:' (containing '@TABLE@'), 'Conjunto de caracteres del archivo:' (set to 'utf-8'), and 'Compresión:' (set to 'Ninguna'). A checkbox 'usar esto para exportaciones futuras' is checked. The 'Formato:' section shows a dropdown menu with 'SQL' selected, and a list of other formats including PDF, which is highlighted by the mouse. The list includes: SQL, CodeGen, CSV, CSV para datos de MS Excel, Microsoft Word 2000, JSON, LaTeX, Tabla MediaWiki, Hoja de cálculo Open Document, Texto Open Document, PDF, Array PHP, SQL, Texto Texy!, Cuaderno XLS Excel 97-2003, Cuaderno XLSX Excel 2007, XML, and YAML.

Para ello, abriremos la tabla en cuestión, y a la hora de exportar, deberemos seleccionar la opción PDF. Trascurridos unos segundos, phpMyAdmin nos habrá generado un fichero, el cual podremos descargar.

MOTORES DE LA BASE DE DATOS:

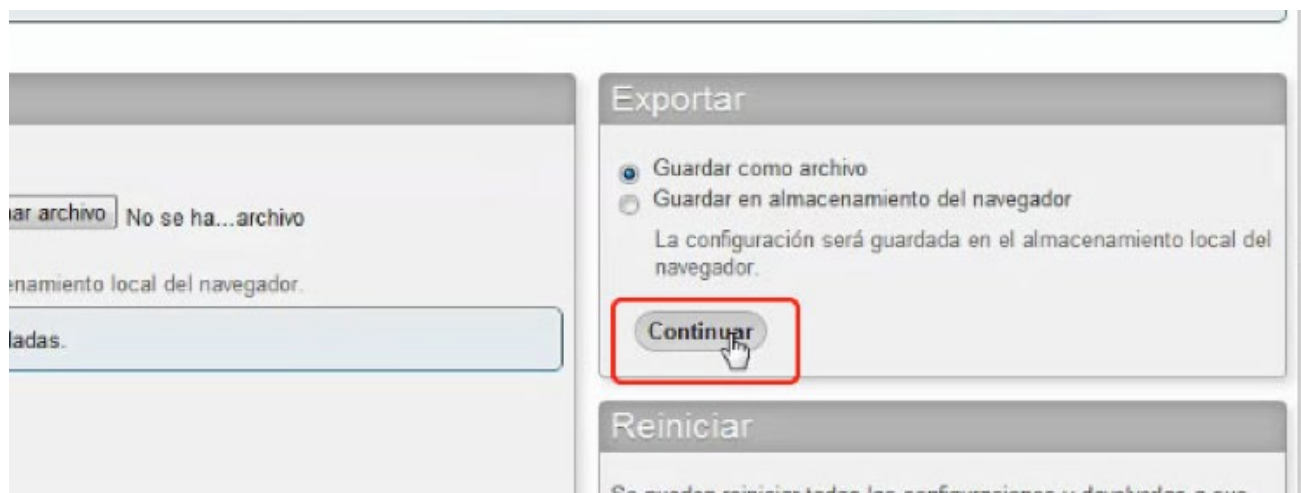
Cada vez que creamos una base de datos, el sistema nos permite elegir que motor utilizar. Por norma general, la opción más adecuada será **InnoDB**, pero hay casos concretos en los que debamos utilizar otro motor. Si queremos visualizar los motores de la base de datos disponibles, deberemos acceder a la pestaña **Motores**.



Nos aparecerá un listado de todos los motores con una breve descripción. Si queremos más información, podremos pulsar sobre el propio nombre del motor. Con ello accederemos a la página de MySQL donde se nos explica de forma más extendida las características de dicha opción.

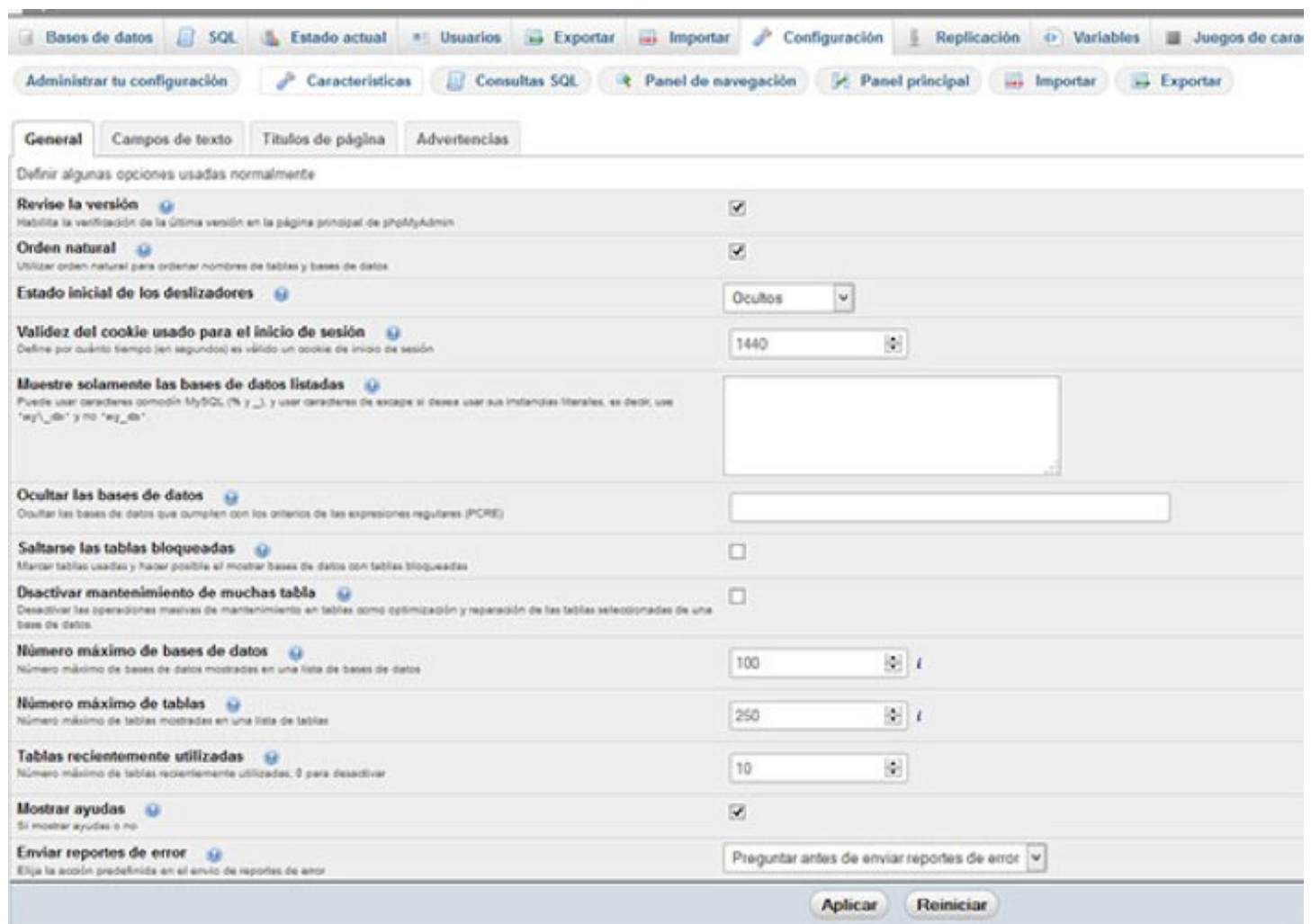
CONFIGURACIÓN DEL SERVIDOR:

Dentro de la pestaña **Configuración** disponemos de una serie de parámetros los cuales podremos personalizar. Como tenemos la posibilidad, es conveniente antes de realizar cualquier cambio, efectuar una copia de seguridad de la configuración del servidor.



Podemos realizar la copia de seguridad de dos formas: o bien descargando un fichero de configuración para poder subirlo cuando sea necesario, o bien, realizar un almacenamiento temporal en el navegador. Hay que tener en cuenta que esta segunda opción se perderá en el momento en el que cerremos el navegador.

A continuación, podremos acceder a las opciones de configuración de phpMyAdmin donde podremos personalizar los aspectos visuales, la configuración o el comportamiento de la aplicación. De entre las muchas opciones que tenemos disponibles, nombramos diversos aspectos como la utilización de AJAX, el time out de la sesión, limitar el número de bases de datos o de tablas, aparición del logo de phpMyAdmin, parámetros por defecto al exportar tablas, etc.



The screenshot shows the 'Configuración' (Configuration) page in phpMyAdmin. The top navigation bar includes links for 'Bases de datos', 'SQL', 'Estado actual', 'Usuarios', 'Exportar', 'Importar', 'Configuración', 'Replicación', 'Variables', and 'Juegos de caracteres'. Below this is a sub-navigation bar with 'Administrar tu configuración', 'Características', 'Consultas SQL', 'Panel de navegación', 'Panel principal', 'Importar', and 'Exportar'. The main content area is titled 'General' and contains various settings:

- Revisar la versión:** A checkbox that is checked, with a description: 'Habilita la verificación de la última versión en la página principal de phpMyAdmin'.
- Orden natural:** A checkbox that is checked, with a description: 'Utilizar orden natural para ordenar nombres de tablas y bases de datos'.
- Estado inicial de los deslizadores:** A dropdown menu set to 'Ocultos'.
- Validez del cookie usado para el inicio de sesión:** A text input field containing '1440', with a description: 'Define por cuánto tiempo (en segundos) es válido un cookie de inicio de sesión'.
- Muestre solamente las bases de datos listadas:** A text input field with a description: 'Puede usar caracteres comodín MySQL (% y _) y usar caracteres de escape si desea usar sus instancias literales, es decir, use \'aj_db\' y no \'aj_db\''.
- Ocultar las bases de datos:** A text input field with a description: 'Ocultar las bases de datos que cumplen con los criterios de las expresiones regulares (PCRE)'.
- Saltarse las tablas bloqueadas:** A checkbox that is unchecked, with a description: 'Marcar tablas usadas y hacer posible el mostrar bases de datos con tablas bloqueadas'.
- Desactivar mantenimiento de muchas tabla:** A checkbox that is unchecked, with a description: 'Desactivar las operaciones masivas de mantenimiento en tablas como optimización y reparación de las tablas seleccionadas de una base de datos'.
- Número máximo de bases de datos:** A text input field containing '100', with a description: 'Número máximo de bases de datos mostrados en una lista de bases de datos'.
- Número máximo de tablas:** A text input field containing '250', with a description: 'Número máximo de tablas mostrados en una lista de tablas'.
- Tablas recientemente utilizadas:** A text input field containing '10', with a description: 'Número máximo de tablas recientemente utilizadas, 0 para desactivar'.
- Mostrar ayudas:** A checkbox that is checked, with a description: 'Si mostrar ayudas o no'.
- Enviar reportes de error:** A dropdown menu set to 'Preguntar antes de enviar reportes de error', with a description: 'Elija la acción predefinida en el envío de reportes de error'.

At the bottom of the configuration page are two buttons: 'Aplicar' and 'Reiniciar'.

BIBLIOGRAFÍA:

SQL in 10 Minutes, Sams Teach Yourself

Autor: Ben Forta(author)

País: USA

Editorial: Sams Publishing; 4 editions

Año de publicación: 2012

SQL: Learn SQL In A DAY! - The Ultimate Crash Course to Learning the Basics of SQL In No Time (SQL, SQL Course, SQL Development, SQL Books, SQL for Beginners)

Autor: Acodemy

País: USA

Editorial: CreateSpace Independent Publishing Platform

Año de publicación: September 9, 2015

CIBERGRAFÍA

Fundamentos del MySQL

Autor: Glenn Nieto

Vínculo: <http://elmundodemysql.blogspot.mx/>

Editor: Glenn Nieto

Año de publicación: 08/02/2007

Fundamentos de MySQL

Autor: Jose Vicente Carratala

Vínculo: <https://www.video2brain.com/mx/cursos/mysql>

Editor: Jose Vicente Carratala

Año de publicación: 21/09/2011