Gestiona bases de datos con MySQL

Gerald Kogler

geraldo@servus.at

Instrucciones preliminares

Podéis descargar este documento aquí:
https://github.com/geraldo/curso_mysql/blob/main/mysql_dia1.pdf

Posteriormente hace falta descargar el instalador de XAMPP desde aquí:
https://www.apachefriends.org/download.html

Seguimos los pasos de instalación de XAMPP para el sistema operativo usado. Si usáis Windows entonces recomiendo instalarlo en la carpeta predefinida:
c:\xampp

Por cualquier problema que nos encontramos podemos consultar la ayuda online:
https://www.apachefriends.org/faq_windows.html

Nos presentamos – Curso

En este curso se enseña la creación y administración de bases de datos con MySQL/MariaDB, dominar el lenguaje SQL y conectarlo a una web usando PHP.

Aprenderás instalar y configurar phpMyAdmin para administrar tu base de datos, crear consultas y subconsultas de SQL, como también la importación y exportación de datos.

Entre muchos otros aspectos estudiaremos:

- Introducción a les bases de datos
- Instalación de una base de datos
- Crear una base de datos con phpMyAdmin
- Lenguaje SQL
- Importación y exportación de la base de datos
- Administración de la base de datos
- Consultes SQL de cerca
- Sub-consultas SQL
- Conexión de una base de datos con un sitio web

Nos presentamos - Formador

Gerald Kogler

Ingeniero técnico en Informática de Gestión (Universidad de Linz/Austria)

Tengo 30 años de experiencia trabajando en el sector de desarrollo web, tanto front end como back end. Además de programar webs a medidas en PHP, también desarrollo en symphony, Drupal y Wordpress.

En los últimos años me he especializado en la programación GIS desarrollando visores de mapas web y programando plugins para QGIS.

https://geraldo.github.io
https://github.com/geraldo
https://gitlab.com/geraldo1

También imparto cursos de formación o talleres a medida de OpenLayers, MySL, QGIS, PyQGIS, Python, etc.

Nos presentamos - Asistentes

Para poder sacar el máximo rendimiento del curso sería muy interesante conocer vuestro perfil, motivaciones y inquietudes:

- Nombre
- Formación
- Experiencia profesional
- Conocimientos de bases de datos y SQL
- Conocimientos de lenguajes de programación como PHP o otros
- Qué esperas aprender en estos sesiones?
- Utilices o pienses utilizar MySQL o MariaDB en tu trabajo?
- Para que crees que necesitas aplicar SQL o PHP en la programación web?
- ...

Temario día 1

1 Introducción a MySQL
1.1 Historia y características de MySQL/MariaDB
1.2 Instalación y configuración inicial
1.3 Herramientas de administración (MySQL Shell, Adminer, phpMyAdmin, DBeaver)

2 Estructura y arquitectura de MySQL
2.1 Arquitectura de bases de datos
2.2 Tipos de datos en MySQL
2.3 Creación y gestión de bases de datos y tablas

3 Comandos básicos de SQL
3.1 Comandos SQL básicos SELECT, FROM, WHERE, ORDER BY, LIMIT, HAVING
3.2 Agregación con COUNT(), SUM(), AVG(), MIN(), MAX()
3.3 Comando SQL GROUP BY

1 - Introducción a MySQL

Introducción - Historia y características

MySQL es la base de datos de código abierto más popular del mundo. Junto con **Oracle** y **Microsoft SQL Server** (de código propietario) es una de las bases de datos más populares del mundo para entornos de desarrollo web.

Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation.

Cuando fue comprado por Oracle en 2010 su inventor Michael Widenius hizo un fork denominado **MariaDB** para asegurar que siga existiendo una versión libre de MySQL. Actualmente es la versión utilizado por la mayoría de servidores web bajo Linux.

La versión actual de MySQL es **8.4** (LTS), que tiene soporte hasta abril del 2032.

La versión actual de MariaDB es **11.4** (LTS), que tiene soporte hasta mayo del 2029.

MariaDB tiene una alta compatibilidad con MySQL ya que posee las mismas órdenes, interfaces, API y bibliotecas, siendo su objetivo poder cambiar un servidor por otro directamente.

Introducción – Historia y características

Antes de empezar hace falta instalar MySQL/MariaDB en nuestro ordenador.

En este curso utilizaremos un paquete XAMP, que quiere decir:

X – sistema operativo: Linux (LAMP) o Windows (WAMP)
A – servidor web: Apache
M – base de datos: MySQL
P – lenguaje de programación: PHP

El instalador XAMPP que utilizaremos incluye:

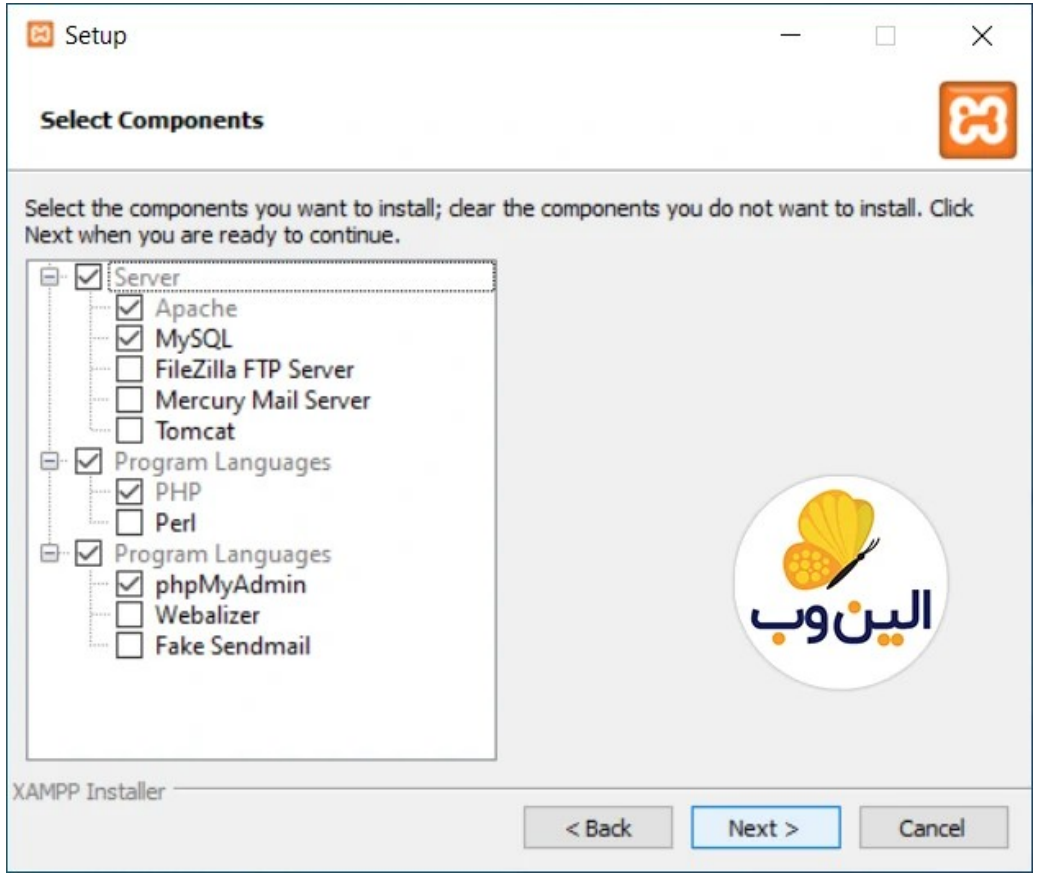
- MariaDB 10
- Apache 2
- PHP 8
- phpMyAdmin 5
- XAMPP Control Panel 3
- etc.

Introducción - Instalación y configuración

Lo descargamos desde aquí: https://www.apachefriends.org/download.html

Seguimos los pasos de instalación de XAMPP para el sistema operativo usado.

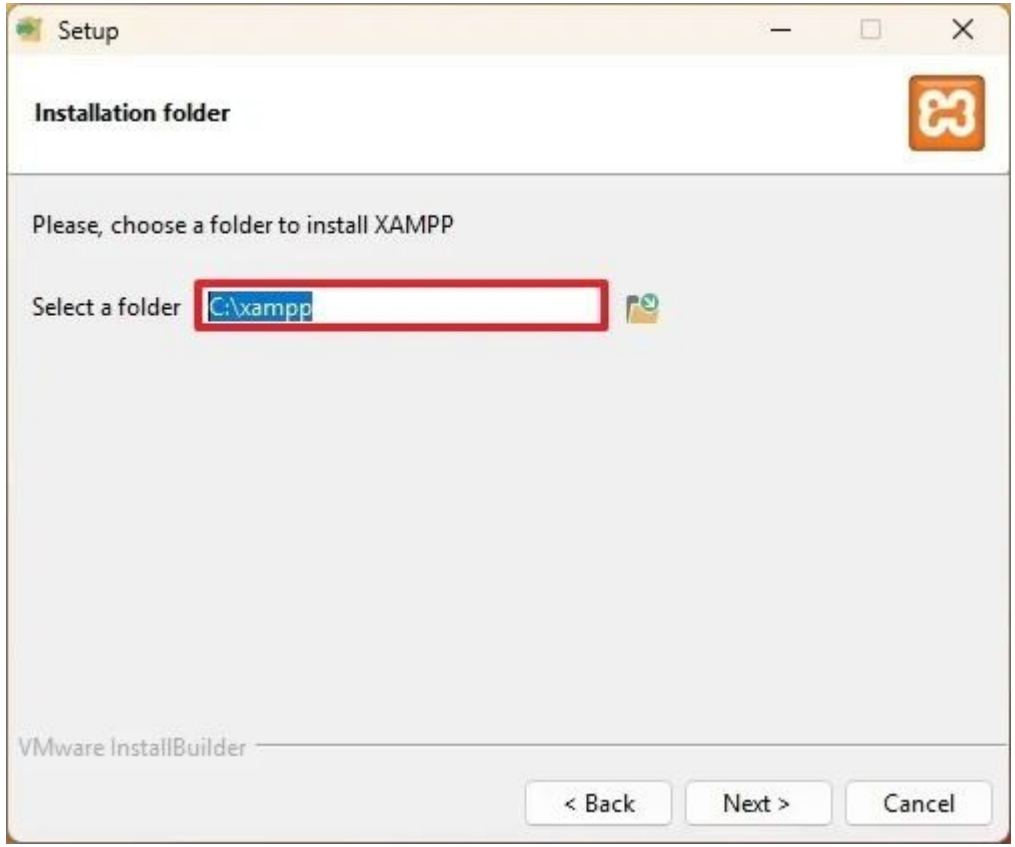
Instalaremos los componentes mínimos para hacer funcionar MySQL, que son Apache, MySQL y phpMyAdmin para administrar nuestra base de datos:



Introducción - Instalación y configuración

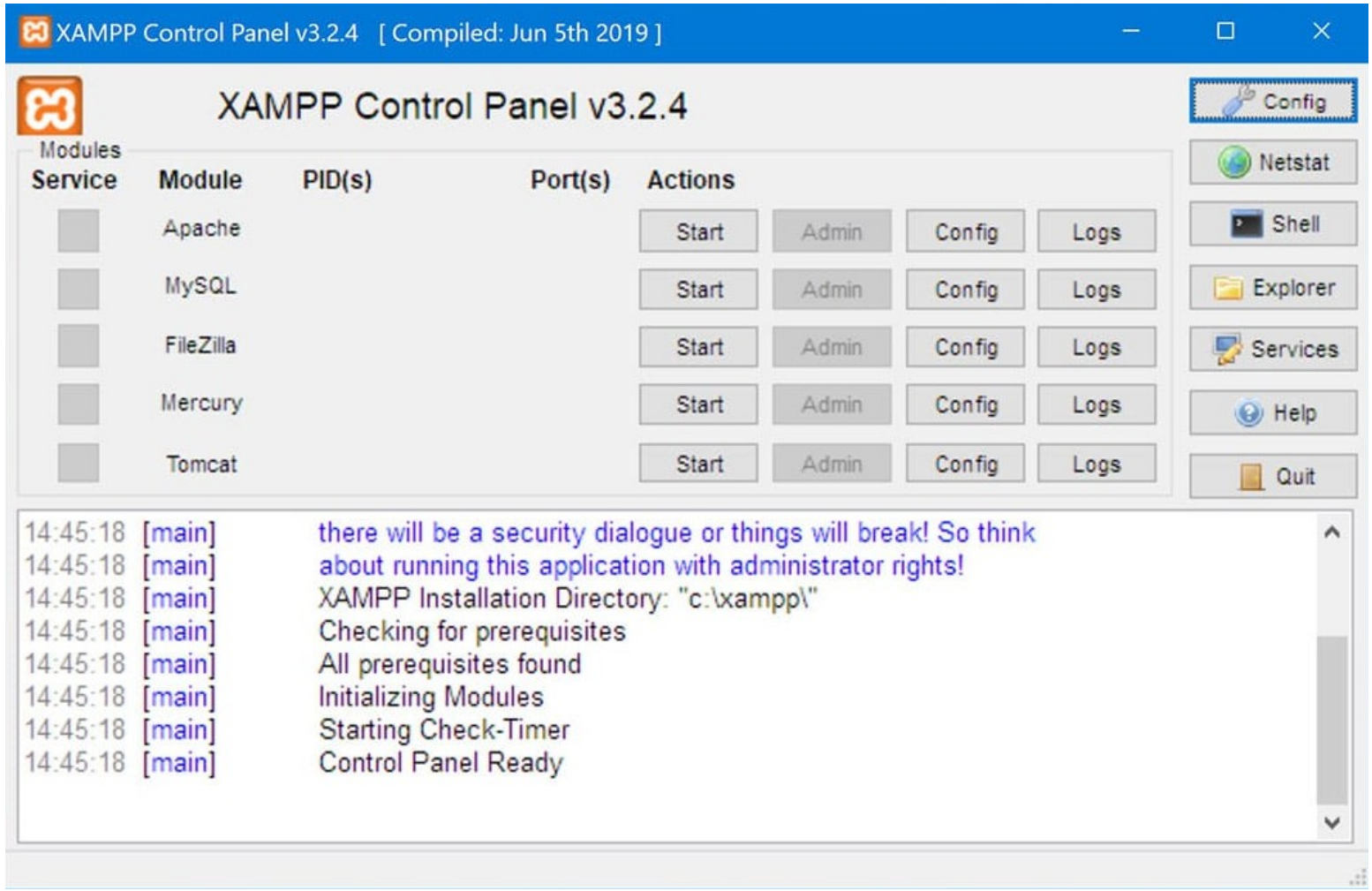
Si usáis Windows entonces recomiendo instalarlo en la carpeta predefinida:
c:\xampp

Por cualquier problema que nos encontramos podemos consultar la ayuda online:
https://www.apachefriends.org/faq_windows.html



Introducción - Instalación y configuración

La gestión de todo el software de XAMPP se hace desde el panel de control de XAMPP:

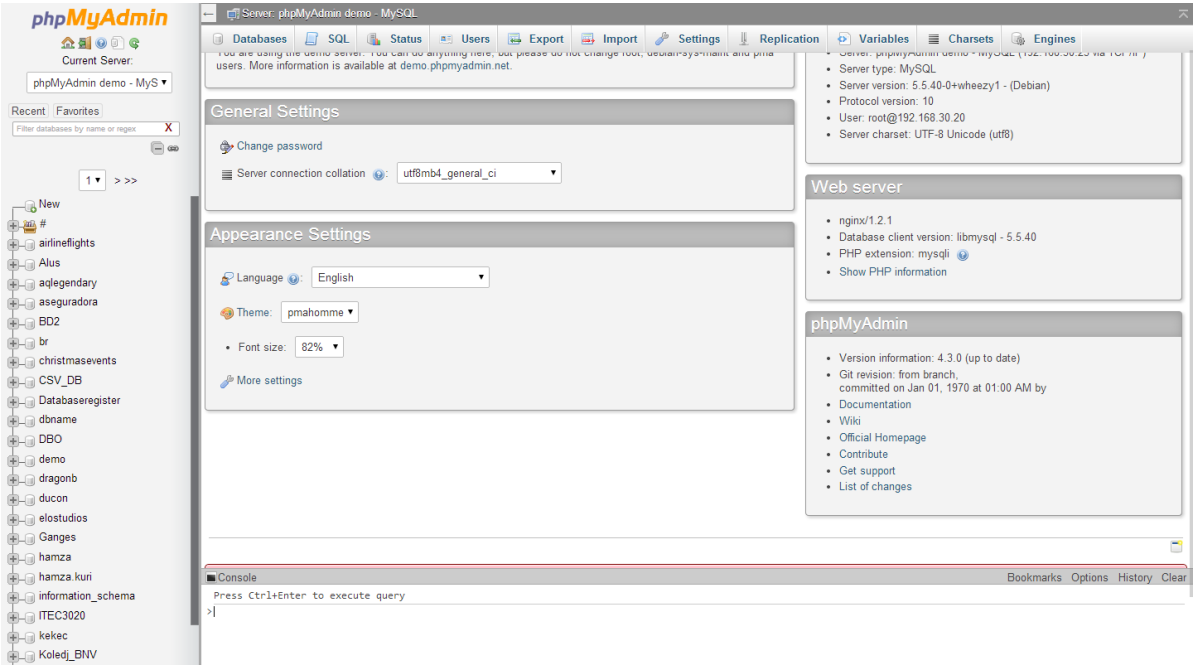


Introducción – Herramientas de administración

Existen una multitud de herramientas de administración para MySQL/MariaDB:

- MariaDB/MySQL Shell: Viene por definición y se gestiona desde la línea de comandos
- Adminer: Gestor ultra sencillo distribuido en un único archivo PHP (https://www.adminer.org/)
- phpMyAdmin: Gestor escrito en PHP muy popular (https://www.phpmyadmin.net/)
- Dbeaver: Herramienta muy potente para gestionar una multitud de bases de datos (https://dbeaver.io/)
- ... (https://en.wikipedia.org/wiki/MySQL#Other_GUI_tools)

En este curso utilizaremos sobre todo **phpMyAdmin**, debido a sus funcionalidades potentes, sencillez de uso y gran popularidad ya que está preinstalado un muchos hostings comerciales.



2- Estructura y arquitectura de MySQL

MySQL – Arquitectura de bases de datos

MySQL es un sistema de gestión de bases de datos relacional, o **RDBMS**, que significa “Relational Database Management System”.

Una base de datos relacional define las **relaciones en forma de tablas**. Las tablas están relacionadas entre si, teniendo un dato en cada tabla en común.

Una **tabla** de una base de datos es una colección de datos que consisten en columnas y filas.

Una **fila** contiene un registro individual de una tabla.

Una **columna** contiene una información específica sobre cada registro de una tabla.

MySQL admite tipos de datos SQL normales en tres categorías principales:

- Tipos numéricos
- Tipos de cadena
- Tipos de fecha y hora

MySQL – Tipos de datos en MySQL

Tipos de datos numéricos

Tipo de datos	Descripción
TINYINT	Un número entero muy pequeño.
SMALLINT	Un número entero pequeño.
MEDIUMINT	Un número entero de tamaño medio.
INT o INTEGRO	Un número entero estándar.
BIGINT	Un número entero grande.
FLOTA	Un número de coma flotante.
DOBLE	Un número de coma flotante de doble precisión.
DECIMAL o NUMÉRICO	Un número en coma fija.

Tipos de datos de fecha y hora

Tipo de datos	Descripción
FECHA	Un valor de fecha en formato AAAA-MM-DD.
TIEMPO	Un valor de tiempo en formato HH:MM:SS.
FECHA	Un valor de fecha y hora en formato AAAA-MM-DD HH:MM:SS.
TIMESTAMP	Un valor de fecha y hora en formato AAAA-MM-DD HH:MM:SS.
AÑO	Un valor de año en formato AAAA o AAAA.

Tipos de datos de cadena

Tipo de datos	Descripción
CHAR	Una cadena de longitud fija.
VARCHAR	Una cadena de longitud variable.
TINYTEXT	Una cadena de texto muy pequeña.
TEXTO	Una pequeña cadena de texto.
MEDIUMTEXT	Una cadena de texto de tamaño medio.
LONGTEXT	Una cadena de texto grande.
ENUM	Un objeto de cadena que sólo puede tener un valor, elegido de una lista de valores predefinidos.
SET	Un objeto de cadena que puede tener cero o más valores, elegidos de una lista de valores predefinidos.

Una explicación detallado de todos los tipos de datos encuentras en
https://www.w3schools.com/mysql/mysql_datatypes.asp

MySQL – Creación y gestión de bases de datos y tablas

Nos imaginamos una tabla para guardar los datos de los **cursos** del Cibernarium:

course_id	course_name	instructor_name	begin_date
1	Gestiona bases de datos con MySQL	Gerald Kogler	2025-01-03
2	Optimización de bases de datos con SQL	Ismael Kale Moyano	2025-01-20

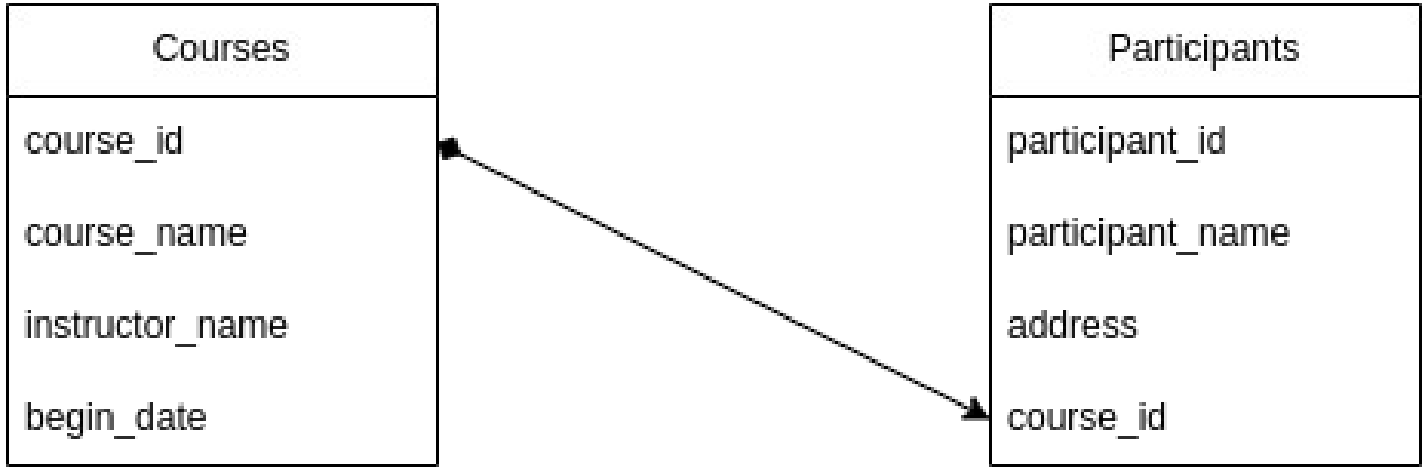
Además queremos guardar los datos de los **participantes**:

participant_id	participant_name	address	course_id
1	Juana Molino	Barcelona	1
2	Antoni Castro	Lleida	1
3	Pedro Masferrer	Girona	1

La **relación** entre las dos tablas la gestionamos a través de la id del curso, que guardamos como referencia en la tabla de participantes.

MySQL – Creación y gestión de bases de datos y tablas

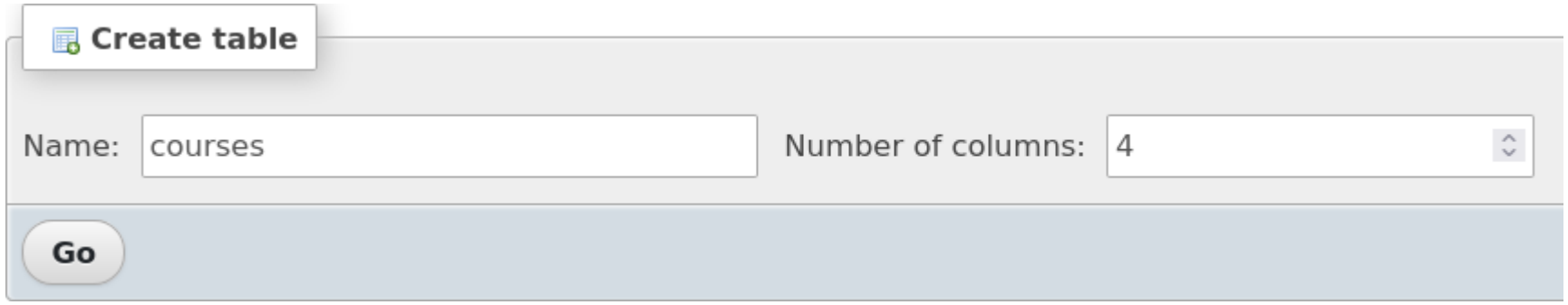
Una forma muy extendida de dibujar las relaciones de las tablas de una base de datos es a través de un **modelo ER** (modelo entidad-relación o entity-relationship model), que es un tipo de **UML** (unified modeling language o lenguaje unificado de modelado). Para el ejemplo de los cursos y participantes, la estructura sería de la siguiente manera:



MySQL – Creación y gestión de bases de datos y tablas

Ahora creamos estas tablas en MySQL utilizando phpMyAdmin. Primero tenemos que crear una base de datos para este ejemplo. La llamamos **cibernarium_courses**.

Ahora dentro de esta base de datos creamos la tabla **courses:**



MySQL – Creación y gestión de bases de datos y tablas

Aquí los parámetros mínimos que definimos para la tabla **courses**:

Server: localhost:3306 » Database: cibernarium_courses » Table: courses

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

Table name: courses Add 1 column(s) Go

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I	Comments
course_id (Pick from Central Columns)	INT					☐	PRIMARY	☐	
course_name (Pick from Central Columns)	VARCHAR	250				☐	---	☐	
instructor_name (Pick from Central Columns)	VARCHAR	250				☐	---	☐	
begin_date (Pick from Central Columns)	DATE					☐	---	☐	

Add index

Index name: PRIMARY

Index choice: PRIMARY

+ Advanced options

Column	Size
course_id [int]	

Preview SQL

Go Cancel

Structure

Table comments:

ngine:

PARTITION definition:

Partition by: (Expression or column list)

Partitions:

Preview SQL Save

MySQL – Creación y gestión de bases de datos y tablas

A así lo veremos en phpMyAdmin una vez creada la tabla **courses**:

Table structure | Relation view

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**course_id**	int			No	*None*			Change Drop More
☐	2	**course_name**	varchar(250)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	3	**instructor_name**	varchar(250)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	4	**begin_date**	date			No	*None*			Change Drop More

La orden de SQL equivalente para crear esta tabla sería:

```
CREATE TABLE courses (
  course_id INT NOT NULL,
  course_name VARCHAR(250) NOT NULL,
  instructor_name VARCHAR(250) NOT NULL,
  begin_date DATE NOT NULL,
  PRIMARY KEY (course_id)
);
```

MySQL – Creación y gestión de bases de datos y tablas

Ahora creamos la segunda tabla **participantes** en MySQL utilizando una orden SQL con la siguiente orden:

```
CREATE TABLE participants (
  participant_id INT NOT NULL,
  participant_name VARCHAR(250) NOT NULL,
  address VARCHAR(250) NOT NULL,
  course_id INT NOT NULL,
  PRIMARY KEY (participant_id)
);
```

En respuesta recibimos la confirmación de la creación de la tabla:

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0182 seconds.)

```
CREATE TABLE participants ( participant_id INT NOT NULL, participant_name VARCHAR(250) NOT NULL, address VARCHAR(250) NOT NULL, course_id INT NOT NULL, PRIMARY KEY (participant_id) );
```

La estructura mostrado con phpMyAdmin es la siguiente:

Table structure | Relation view

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**participant_id**	int			No	*None*			Change Drop More
☐	2	**participant_name**	varchar(250)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	3	**address**	varchar(250)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	4	**course_id**	int			No	*None*			Change Drop More

MySQL – Creación y gestión de bases de datos y tablas

Para que tengamos datos disponibles importaremos ficheros CSV. Se hace desde el menú **Import**, eligiendo el fichero *courses.csv* desde *Browse…* y cambiando solamente el parámetro de formato a **CSV**.

Entonces phpMyAdmin mostrará los datos de la tabla **courses** de la siguiente manera:

←T→	course_id	course_name	instructor_name	begin_date
Edit Copy Delete	1	Gestiona bases de datos con MySQL	Gerald Kogler	2025-01-03
Edit Copy Delete	2	Optimización de bases de datos con SQL	Ismael Kale Moyano	2025-01-20

Haremos lo mismo con los datos de los participantes importanto el fichero participants.csv y veremos el resultado de la siguiente manera:

←T→	participant_id	participant_name	address	course_id
Edit Copy Delete	1	Juana Molino	Barcelona	1
Edit Copy Delete	2	Antoni Castro	Lleida	1
Edit Copy Delete	3	Pedro Masferrer	Girona	1

3 - Comandos básicos de SQL

MySQL – Introducción a SQL

Structured Query Language (SQL) es un lenguaje de programación diseñada para trabajar con bases de datos.

La primera versión es del 1986 y la versión actual del estándar (SQL3) salió en el 1999. La última ampliación es del año 2006.

Muchas bases de datos incorporan sentencies propias al lenguaje base, pero la base funciona para todas. Nosotros haremos servidor algunas funciones propias de la versión de MySQL/MariaDB.

MySQL – SQL Comentarios

Para comentar el código tenemos dos opciones:

```
-- Dos guiones hacen que la línea será un comentario
```

```
/*
Una barra y un asterisco empiezan un comentario
hasta encontrar un asterisco y una barra.
Esto sirve para comentar múltiples línies.
*/
```

MySQL – SQL Select

La instrucció **SELECT** selecciona datos que son enviados al usuario en forma de tabla.

Es la instrucción más básica y la más compleja al mismo tiempo,
ya que contiene muchas variantes y posibles modificaciones.

Al final de cada sentencia tenemos que añadir punto y coma (**;**),
eso sirve para avisar a MySQL que ejecute la sentencia:

```
SELECT 3, 12+2;
```

Show query box

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

```
select 3, 12+2;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

3	12+2
3	14

MySQL – SQL As

En el ejemplo anterior podemos ver que las columnas no tienen nombre. Para cambiar el nombre de las columnas utilizamos **AS**, que le asigna un nombre.

Es una orden para crear nombres o aliases.

```
SELECT 3+2 AS resultado, 6+21 AS respuesta;
```

resultado	respuesta
5	27

MySQL – SQL From

Una base de datos sirve para guardar y leer datos. Vamos a hacer ahora esto último.

FROM nos permite acceder a los datos de una tabla de la base datos. El contenido de las columnas resultantes será ejecutado para todas las filas de la tabla.

```
SELECT * FROM courses;
```

Podemos especificar la columna de cada interacción por su nombre:

```
SELECT course_name FROM courses;
```

También podemos indicar varias columnas:

```
SELECT participant_name, course_id FROM participants;
```

También se pueden hacer operaciones con los datos de cada columna:

```
SELECT course_id, course_id+5 FROM courses;
```

MySQL – SQL Sensibilidad a mayúsculas

El lenguaje SQL solo distingue entre mayúsculas y minúsculas en dos casos:

1. Valores literales (como por ejemplo text)
2. Nombres identificados entre comillas dobles (“)

También ignora las nuevas líneas, espacios extras y tabulaciones.

Hasta ahora hemos mostrado las sentencias con el formato mayúsculas para las instrucciones y minúsculas para los nombres. Esto es debido a motivos históricos. Lo más importante es utilizar siempre el mismo sistema.

Recomendación: Definir todo lo que sea estructural con **minúsculas**:

- nombre de base de datos,
- tablas,
- vistas,
- campos,
- roles,
- etc...

MySQL – SQL Where

La instrucción WHERE introduce **condicionales** al SELECT. Solo se devuelven las filas que cumplan la condición, esta puede ser cualquier expresión que sea de tipo booleano.

```
SELECT columna
        FROM basic.taula
        WHERE columna > 3
;
```

Operaciones booleanas:

Operación	Definición
a=b	Igual que
a <> b a != b	Diferente que
a>b	Más grande que
a >= b	Más grande o igual que
a<b	Más pequeño que
a <= b	Más pequeño o igual que
NOT a	a (álgebra de booleano)
a OR b	a·b (álgebra de booleano)
a AND b	a+b (álgebra de booleano)

El listado completo de operaciones encuentras aquí: https://www.w3schools.com/mysql/mysql_operators.asp

MySQL – SQL Ejercicio Where

Importamos el fichero **countries.csv** a nuestra base de datos **cibernarium_courses** para un pequeño ejercicio. Eso crea la tabla **countries** y importa todos los países del mundo como registros.

Nota: Los ficheros **.ods** son otra alternativa de importación. Son hojas de cálculo de LibreOffice y (por lo menos en Linux) comparado con el .csv tiene la ventaja que phpMyAdmin reconoce la primera fila como cabecera. La forma habitual de importar ficheros es por SQL, esa opción veremos un poco más adelante.

Resuelve los siguientes tareas empleando SELECT con WHERE:

- ¿Cuantos países hay en el mundo?
- ¿Qué código ISO tiene Georgia?
- ¿Qué población tiene Siria?
- ¿Qué PIB por persona tiene Alemania?
- ¿Cuántos países tienen más 100.000.000 de habitantes?
- ¿Cuántos países tienen África?

MySQL – SQL Select In

El operador **IN** te deja especificar múltiples valores en una cláusula WHERE. Es una forma corta de crear múltiples condiciones combinados con OR:

```
SELECT *
    FROM municipis
    WHERE nomcomar in ('Anoia','Penedès')
;
```

MySQL – SQL Select Distinct

SELECT DISTINCT devuelve todos los valores únicos o diferentes de una tabla.

Dentro de una columna muchas veces se repiten valores, pero a veces queremos solamente los valores únicos:

```
SELECT DISTINCT(CONTINENT)
    FROM countries
;
```

MySQL – SQL Order By

ORDER BY se pone después del WHERE y ordena las filas según los campos y criterios elegidos.

```
SELECT NAME_ES, POP_EST AS población
     FROM countries
     WHERE CONTINENT="Europe"
     ORDER BY población
;
```

```
SELECT NAME_ES, GDP_MD AS PIB
     FROM countries
     WHERE CONTINENT="Europe"
     ORDER BY PIB DESC, NAME_ES;
```

Notas:

- Por defecto ORDER BY es ascendente. Aunque se puede utilizar ASC para ser explícito.
- Caracteres y textos serán ordenados alfabéticamente.

MySQL – SQL Limit

LIMIT hace que el resultado final devuelva las primeras filas del resultado que tendría si no le ponemos límit.

Esto permite hacer una carga parcial de una tabla muy grande, y hacer testeo y pruebas con una muestra parcial.

```
SELECT NAME_ES, POP_EST AS población
        FROM countries
        WHERE CONTINENT="Europe"
        ORDER BY población DESC
        LIMIT 10
;
```

Pequeño ejercicio:

- ¿Cuáles son las 5 países más poblados del mundo?
- ¿Cuáles son los 10 países de Europa con menos habitantes?

MySQL – Agregación

La agregación consiste en utilizar más de una fila para calcular un resultado final.

Los casos más habituales son.:

- **SUM()**: Sumatorios
- **COUNT()**: Contar filas
- **AVG(), MIN(), MAX()**: Cálculos estadísticos como por ejemplo la media, el mínimo o el máximo.

Cuando se utiliza la agregación hay que estar alerta con los campos que se agregan (si se intenta utilizar una columna sin agregar, no sabrá que valor coger).

Cuando se hace una agregación, las instrucciones WHERE, ORDER BY, LIMIT, etc. afectan a los datos antes de agregarse.

```
SELECT SUM(valor)
FROM ejemplo
```

Pequeño ejercicio:

- ¿Calcula la suma de habitantes de Europa?
- ¿Cuantos países en el mundo tienen más que un billón de habitantes?
- ¿Cual es la población media de todos los países del mundo?

MySQL – SQL Group By

Agrupa las columnas en grupos, definidos por las columnas deseadas.

Todas las columnas de la tabla final han de estar definidas en los grupos, o ser una función agregativa o bien no ser utilizadas.

```
SELECT CONTINENT, COUNT(*)
        FROM countries
        GROUP BY CONTINENT
;
```

Pequeño ejercicio:

- ¿Cuantos países tiene cada continente?
- ¿Cual es la población media de todos los países por continente?

MySQL – SQL Having

Como ya hemos visto, WHERE afecta a los valores antes de la agregación. Para hacerlo después se puede utilizar **HAVING**.

En el siguiente ejemplo limitamos los continentes para cuales calculamos la población media por la cantidad de países con un mínimo de dos:

```
SELECT CONTINENT, COUNT(*), AVG(POP_EST)
        FROM countries
        GROUP BY CONTINENT
        HAVING COUNT(*)>1
;
```

MySQL – SQL Ejercicios

Importamos el fichero SQL **municipis.sql** que contiene todos los municipios de Cataluña.

Resuelve los siguientes tareas empleando los diferentes ordenes SQL que aprendimos:

1. ¿Qué id tiene la comarca del Bages?
2. ¿Cuántos municipios hay en la comarca del Baix Llobregat?
3. ¿Cuántos municipios tienen más de 1000 metros de altitud y más de 1000 habitantes?
4. ¿Cuántos municipios tiene más de 200 km2?
5. ¿Cuáles son los 10 municipios de Cataluña con más habitantes ordenados con el más poblado primero?
6. ¿Cuáles son las 5 comarcas de Cataluña con más área (en km^2) ordenados con la más grande primera?
7. ¿Cuántos municipios hay en cada comarca?
8. ¿Cuántos habitantes hay en cada provincia de Cataluña?

MySQL – SQL Ejemplo SQL Simple

Aquí un ejemplo de SELECT simple con todo lo que hemos visto. Lo interesante en este ejemplo es revisar el orden de las funciones:

```
SELECT codiprov , sum(areapol) as area
        FROM municipis
        WHERE altitud < 1000
        GROUP BY codiprov
        HAVING codiprov IN ('08','17', '25')
        ORDER BY area DESC
        LIMIT 2
;
```

Así es el orden como MySQL lo ejecuta:

Espero que os ha agradado esta sesión.
Muchas gracias por vuestra asistencia.

Gerald Kogler

geraldo@servus.at

https://geraldo.github.io
https://github.com/geraldo
https://gitlab.com/geraldo1