

Gestiona bases de datos con MySQL

Día 2

Gerald Kogler

geraldo@servus.at

Instrucciones preliminares

Podéis descargar este documento aquí:

https://github.com/geraldo/curso_mysql_online/blob/main/mysql_dia2.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

Temario día 2

4 Herramientas de administración alternativas

5 Agregaciones y agrupaciones

5.2 Agregación con COUNT(), SUM(), AVG(), MIN(), MAX()

5.3 Comando SQL GROUP BY

6 Comandos de SQL para gestionar MySQL

6.1 INSERT, UPDATE, DELETE

6.2 CREATE, DROP

4 – Herramientas de administración

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)

Para conectar desde cualquier cliente a nuestra base de datos MySQL, necesitamos los siguientes datos:

- Server Host: *localhost*
- Port: *3306*
- Username: *root*
- Password: *[mysql no tiene contraseña por definición en XAMPP]*

Herramientas de administración

Ahora usaremos **DBeaver** para conectarnos a nuestra base de datos *cibernarium_courses*:

Connection "localhost" configuration

MySQL connection settings

Connection settings

Initialization

Shell Command

Client identification

Transactions

General

Metadata

Errors and timeouts

Data Transfer

Data Editor

SQL Editor

Main Driver properties SSH SSL + Network configurations...

Server

Connect by: ☒ Host ☐ URL

URL: jdbc:mysql://localhost:3306/cibernarium_courses

Server Host: localhost Port: 3306

Database: cibernarium_courses

Authentication (Database Native)

Username: root

Password: ☒ Save password

Advanced

Server Time Zone: Auto-detect

Local Client: /usr

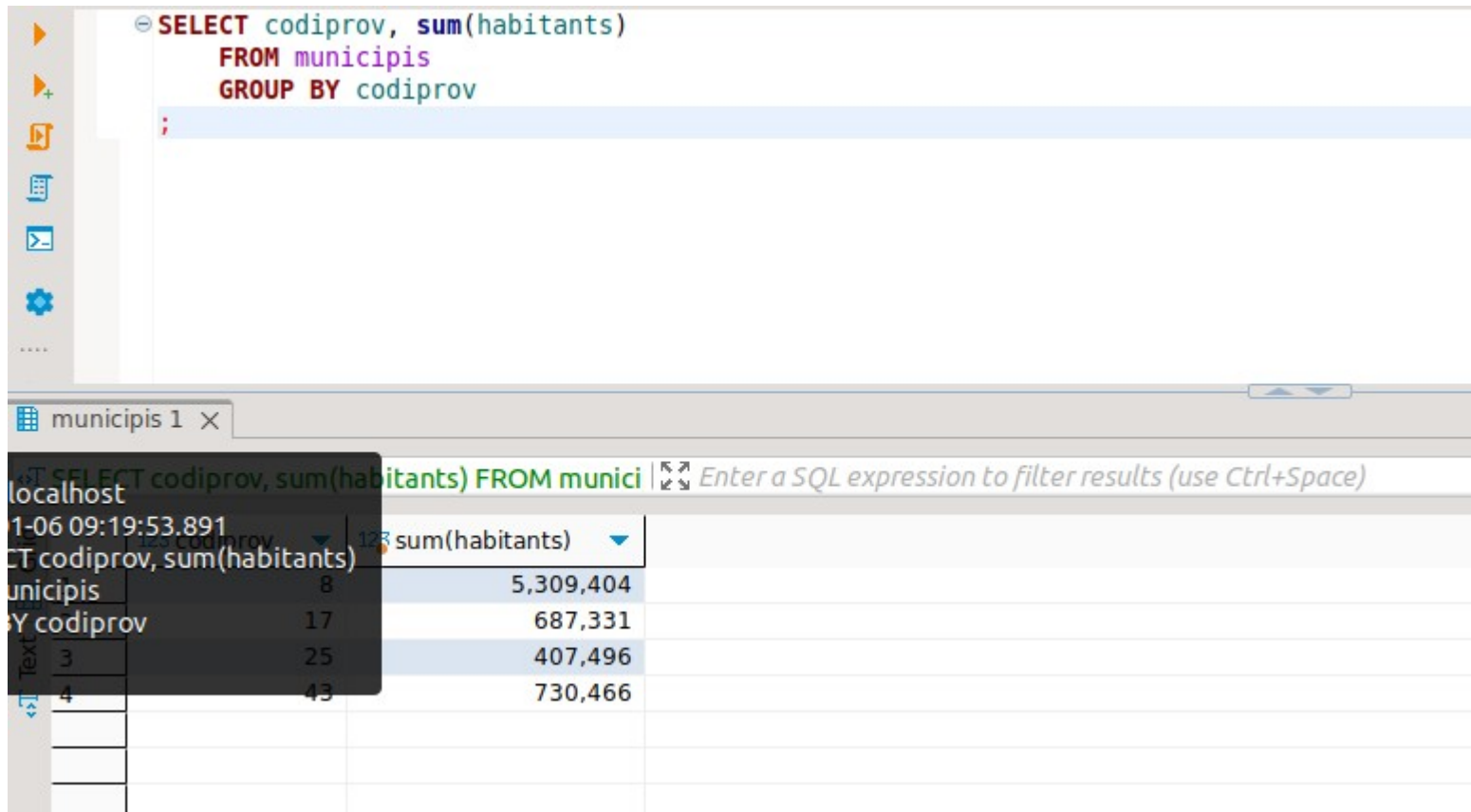
[Connection variables information](#) [Database documentation](#)

Driver name: MySQL Driver Settings Driver license

Test Connection ... Cancel OK

Herramientas de administración

DBeaver tiene su propio **SQL Editor** para ejecutar consultas de SQL:



The screenshot displays the DBeaver SQL Editor interface. The top panel shows a SQL query:

```
SELECT codiprov, sum(habitants)
FROM municipis
GROUP BY codiprov
;
```

The bottom panel shows the results of the query in a table format. The table has two columns: 'codiprov' and 'sum(habitants)'. The results are as follows:

codiprov	sum(habitants)
8	5,309,404
17	687,331
25	407,496
43	730,466

Herramientas de administración

Ahora utilizaremos **DBeaver** para resolver los ejercicios que nos quedaron pendientes de la primera clase.

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 km²) ordenados con la más grande primera?
7. ¿Cuántos municipios hay en cada comarca?
8. ¿Cuántos habitantes hay en cada provincia de Cataluña?

5 - Agregaciones y agrupaciones

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 límite.

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?
- ¿Cuántos 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

Agrupar 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:

- ¿Cuántos países tiene cada continente?
- ¿Cuál 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 km²) 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:



6 - Comandos de SQL para gestionar MySQL

MySQL – SQL Alter Table

Utilizaremos ALTER TABLE para modificar la estructura de una tabla existente. Podemos tanto modificar una columna existente como añadir una nueva.

Añadimos una columna nueva que contenga el teléfono de los participantes de los cursos:

```
ALTER TABLE participantes  
ADD COLUMN telefono varchar(20);
```

Para borrar una columna hacemos lo siguiente:

```
ALTER TABLE participantes  
DROP COLUMN telefono;
```

MySQL – SQL Alter Table

Para definir la id como auto-incrementar, hacemos lo siguiente:

```
ALTER TABLE participantes  
MODIFY COLUMN participante_id INT NOT NULL AUTO_INCREMENT;
```

Para dar un valor por defecto a la columna *direccion*, hacemos lo siguiente:

```
ALTER TABLE participantes  
MODIFY COLUMN direccion VARCHAR(250) DEFAULT NULL;
```

Para definir *curso_id* como índice de otra tabla, hacemos lo siguiente:

```
ALTER TABLE participantes  
ADD CONSTRAINT fk_curso_id FOREIGN KEY (curso_id) REFERENCES cursos(curso_id);
```

Ejercicio:

- Añadimos una columna nueva a la tabla de municipios nombrada *nomprov* para guardar el nombre de las provincias

MySQL – SQL Constraints

Podemos definir reglas como se insertan los datos en las columnas usando los siguientes limitadores:

- NOT NULL - Aseguramos que la columna no puede ser NULL.
- UNIQUE - Aseguramos que cada valor de la columna es único.
- PRIMARY KEY - Es una combinación de NOT NULL y UNIQUE. Identifica cada valor de la columna.
- FOREIGN KEY - Evita destruir enlaces entre tablas.
- CHECK - Asegura que la condición de la columna cumpla una cierta condición.
- DEFAULT - Define un valor predefinido en caso que no haya valor al insertar.
- CREATE INDEX - Usado para crear y conseguir datos de una tabla muy rápido.

MySQL – SQL Insert

Sirve para añadir filas a una tabla. Básicamente hay dos formas de llamar al INSERT:

1. Especificar nombres de columnas y valores a insertar:

```
INSERT INTO table_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);
```

2. Si vas a insertar valores para todas las columnas, entonces no hace falta nombrarlas. En este caso hay que asegurar que el orden de los valores es igual al orden de columnas en la tabla:

```
INSERT INTO table_name  
VALUES (value1, value2, value3, ...);
```

En este caso si hace falta que las columnas para los valores que no vamos a insertar tengan un valor por defecto.

MySQL – SQL Insert

Para probar insertamos nuevos registros a la tabla de participantes de cursos:

```
INSERT INTO participantes  
VALUES (4, "Ana", "Tarragona", 2);
```

En el caso que no queremos rellenar todas las columnas - y incluso podemos cambiar el orden de valores a insertar) – tenemos que definir los nombres de columnas de la siguiente manera:

```
INSERT INTO participantes (curso_id, nombre)  
VALUES (2, "Joan");
```

Ejercicios:

- Añadimos un nuevo curso definiendo los valores de todas las columnas.
- Añadimos un nuevo curso solamente definiendo el nombre y el instructor.

MySQL – SQL Update

UPDATE actualiza valores de la tabla:

```
UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

Para modificar la dirección de una participante de curso hacemos lo siguiente:

```
UPDATE participantes  
SET direccion="Tarragona"  
WHERE participante_id=5;
```

Cuidado con UPDATE! En caso de usarlo sin un WHERE todos los registros se sobrescriben!

Ejercicios:

- Actualizamos la tabla *municipis* insertando todos los nombres de las provincias a la columna *nomprov* utilizando los códigos oficiales de <https://www.idescat.cat/codis/?id=50&n=11&lang=es>

MySQL – SQL Delete

Elimina filas o registros de la tabla según el criterio elegido. Si no hay criterio, elimina todos los valores de la tabla (pero no la tabla en si).

```
DELETE FROM table_name  
WHERE condition;
```

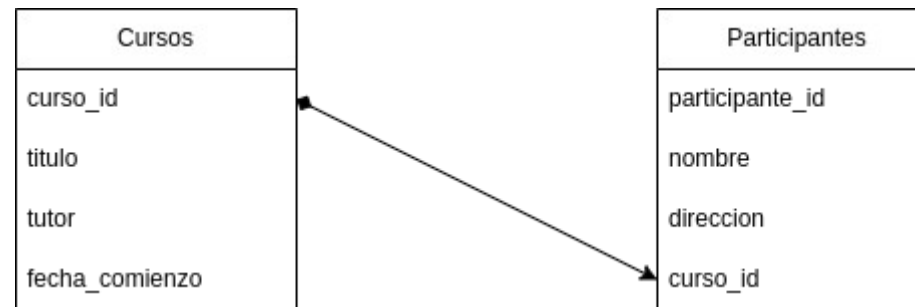
Para borrar un participante de un curso, hagamos lo siguiente:

```
DELETE FROM participantes  
WHERE participante_id=6;
```

MySQL – SQL Create

Aquí repetimos la orden de crear una tabla, ahora con varios constraints definidos:

```
CREATE TABLE participantes (  
    participante_id INT NOT NULL AUTO_INCREMENT,  
    nombre VARCHAR(100) NOT NULL,  
    direccion VARCHAR(100) DEFAULT NULL,  
    curso_id INT NOT NULL FOREIGN KEY (curso_id),  
    PRIMARY KEY (participante_id),  
    FOREIGN KEY (curso_id) REFERENCES cursos(curso_id)  
);
```



En caso que ya existe una tabla con el mismo nombre, CREATE no se ejecuta. Si queremos sobrescribir una tabla existente, podemos escribir CREATE OR REPLACE.

MySQL – SQL Drop

DROP TABLE sirve para eliminar una tabla completa, incluyendo estructura y datos:

```
DROP TABLE table_name;
```

En el caso que queremos eliminar todos los datos, pero no la tabla misma, usamos TRUNCATE TABLE:

```
TRUNCATE TABLE table_name;
```

Para borrar toda una base de datos, usamos DROP DATABASE:

```
DROP DATABASE databasename;
```

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

Gerald Kogler

gerald@servus.at

<https://gerald.github.io>

<https://github.com/gerald>

<https://gitlab.com/gerald1>