SQL

Definición de SQL

SQL (Structured Query Language) es un lenguaje de programación diseñado para gestionar y manipular bases de datos relacionales. Permite:

- Crear, modificar y eliminar estructuras de datos (tablas, índices, etc.).
- Insertar, actualizar, consultar y borrar registros.
- Controlar el acceso a los datos y garantizar su integridad.
- https://blog.ansi.org/sql-standard-iso-iec-9075-2023-ansi-x3-135/
- https://dev.mysql.com/doc/refman/8.4/en/compatibility.html

Características clave

- Estándar internacional (ANSI/ISO).
- Compatible con sistemas como MySQL, PostgreSQL, Oracle y SQL Server.
- Lenguaje declarativo: se especifica *qué* hacer, no *cómo* hacerlo.

Que puede hace SQL?

- SQL puede ejecutar consultas en una base de datos.
- SQL puede recuperar datos de una base de datos.
- SQL puede insertar registros en una base de datos.
- SQL puede actualizar registros en una base de datos.
- SQL puede eliminar registros de una base de datos.
- SQL puede crear nuevas bases de datos.
- SQL puede crear nuevas tablas en una base de datos.
- SQL puede crear procedimientos almacenados en una base de datos.
- SQL puede crear vistas en una base de datos.
- SQL puede establecer permisos sobre tablas, procedimientos y vistas.

Usos

1. Desarrollo de aplicaciones:
    1. Backend de sistemas (ej: sitios web, apps móviles).
    2. Integración con lenguajes como Python, Java o PHP.
2. Análisis de datos:
    1. Generación de reportes y dashboards.
    2. Minería de datos y Business Intelligence (BI).
3. Administración de bases de datos:
    1. Optimización de consultas.
    2. Respaldo y recuperación de datos.
4. Automatización de procesos:
    1. Scripts para migración de datos o ETL (Extract, Transform, Load).

RDBMS

RDBMS significa Sistema de Gestión de Bases de Datos Relacionales (Relational Database Management System).

El RDBMS es la base de SQL y de todos los sistemas modernos de bases de datos, como MS SQL Server, IBM DB2, Oracle, MySQL y Microsoft Access.

En un RDBMS, los datos se almacenan en objetos de la base de datos llamados tablas. Una tabla es una colección de entradas de datos relacionados y está formada por columnas (campos) y filas (registros).

Qué se puede hacer con SQL

Operaciones Básicas (CRUD)

Operación	**Sentencia SQL**	**Ejemplo en MySQL**
Crear	CREATE TABLE	CREATE TABLE usuarios (id INT PRIMARY KEY, nombre VARCHAR(50));
Leer	SELECT	SELECT * FROM usuarios WHERE edad > 25;
Actualizar	UPDATE	UPDATE usuarios SET email = 'nuevo@mail.com' WHERE id = 101;
Eliminar	DELETE	DELETE FROM usuarios WHERE estado = 'inactivo';

Tipos de Sentencias SQL

DDL (Data Definition Language)

DDL significa Data Definition Language o Lenguaje de Definición de Datos, en español. Este lenguaje permite definir las tareas de las estructuras que almacenarán los datos.

- CREATE: Crear tablas, bases de datos.
- ALTER: Modificar tablas (ej: añadir columnas).
- DROP: Eliminar tablas.
- TRUNCATE: Vaciar una tabla (elimina todos los registros).
- COMMENT: Utilizado para agregar comentarios al diccionario de datos.
- RENAME: Tal como su nombre lo indica es utilizado para renombrar objetos.

Ejemplo:

sql
ALTER TABLE empleados
ADD COLUMN fecha_ingreso DATE DEFAULT '2023-01-01';

DML (Data Manipulation Language)

DML significa Data Manipulation Language o Lenguaje de Manipulación de Datos, en español. Este lenguaje permite realizar diferentes acciones a los datos que se encuentran en una base de datos.

Permite recuperar, almacenar, modificar, eliminar, insertar y actualizar datos de una base de datos.

- SELECT
- INSERT
- UPDATE
- DELETE

DCL (Data Control Language)

Permite crear roles, permisos e integridad referencial, así como el control al acceso a la base de datos.

- GRANT: Otorgar permisos.
- REVOKE: Revocar permisos.

Ejemplo:

sql
GRANT SELECT, INSERT ON base_datos. TO 'usuario_app'@'localhost';*

TCL (Transaction Control Language)

Administra transacciones:

- COMMIT: Guardar cambios.

- ROLLBACK: Utilizado para deshacer la modificación que hice desde el último COMMIT.
- SAVEPOINT: Punto de rollback parcial.

Sentencias mas usadas en SQL

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

https://www.w3schools.com/sql/sql_intro.asp

MySQL

Tipos de datos

Los tipos de datos se utilizan para determinar los valores que puede contener una columna. Funciona como una especie de metadatos que ayudan a SQL a entender qué tipo de datos esperar en cada columna y cómo procesar las consultas contra una columna concreta.

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

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

https://dev.mysql.com/doc/refman/8.4/en/data-types.html

En las tablas siguientes puedes encontrar una visión general de los principales tipos de datos de 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 INTEGER	Un número entero estándar.
BIGINT	Un número entero grande.
FLOAT	Un número de coma flotante.
DOBLE	Un número de coma flotante de doble precisión.
DECIMAL o NUMERIC	Un número en coma fija.

https://dev.mysql.com/doc/refman/8.4/en/numeric-types.html

Tipos de datos de fecha y hora

Tipo de datos	Descripción
DATE	Un valor de fecha en formato AAAA-MM-DD.
TIME	Un valor de tiempo en formato HH:MM:SS.
DATETIME	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.
YEAR	Un valor de año en formato AAAA o AAAA.

https://dev.mysql.com/doc/refman/8.4/en/date-and-time-types.html

Tipos de datos de cadena

Tipo de datos	Descripción
CHAR	Una cadena de longitud fija.
VARCHAR	Una cadena de longitud variable.
BINARY	Una cadena binaria.
VARBINARY	Una cadena binaria de longitud variable.
BLOB	Es un objeto binario de gran tamaño que puede contener una cantidad variable de datos. Como archivos multimedia, word, pdf, xml, zip, xls...
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.

https://dev.mysql.com/doc/refman/8.4/en/string-types.html

Acceder a MySQL

Una vez que hayas instalado MySQL en tu ordenador, puedes empezar a utilizarlo desde tu terminal. Para lanzar MySQL, puedes acceder a él con tu nombre de usuario y su contraseña asociada. En este caso, accederemos a MySQL con la cuenta root, junto con la contraseña de root que establecimos durante la instalación. Puesto que estamos iniciando sesión en la misma máquina en la que se ejecuta MySQL, no necesitamos proporcionar información en el parámetro host (-h):

> mysql -uroot -p

Si quieres añadir nuevos usuarios, escribe:

> CREATE USER 'username' IDENTIFIED BY 'password';

Una vez que hayas creado un nuevo usuario, puedes entrar en MySQL utilizando este comando y se mostrará un mensaje de bienvenida, seguido de un prompt *mysql>:*

mysql -u username -p
Welcome to the MySQL monitor. Commands end with; or \g.
Your MySQL connection id is 25
Server version: 8.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
Owners.

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

mysql>

Cómo crear una base de datos MySQL

Es hora de crear tu primera base de datos en MySQL. Para este tutorial, queremos crear una base de datos llamada "unicatolica", que contendrá información sobre algunos programas de unicatolica. La base de datos se aloja localmente.

Para crear una base de datos, utiliza el siguiente comando:

mysql> CREATE DATABASE unicatolica;

Para comprobar que la base de datos está creada, escribe lo siguiente y se mostrará una tabla con todas las bases de datos disponibles:

mysql> SHOW databases;
+--------------------+
| Database |
+--------------------+
| unicatolica |
| mysql |
+--------------------+
5 rows in set (0,00 sec)

Ahora ya puedes acceder a tu nueva base de datos.

mysql> USE unicatolica;
Database changed

Crear una tabla

Hasta ahora, has creado tu primera base de datos. Pero sigue vacía. Ahora vamos a crear la primera tabla de la base de datos. Queremos crear una tabla llamada 'cursos', con información sobre varios programas disponibles en el catálogo de programas de unicatolica.

Es hora de crear tu primera tabla en la base de datos "unicatolica". Queremos crear una tabla con información sobre algunos de los programas del Catálogo de programas unicatolica.

La clave/llave principal de la tabla debe ser programa_id (observa que sólo ésta está en negrita), y su tipo de datos debe ser un entero. Una clave primaria es una restricción que obliga a que los valores de las columnas no sean nulos y sean únicos. Te permite identificar de forma única una instancia concreta o un conjunto de instancias presentes en la tabla.

Las columnas restantes proporcionan información sobre el nombre del curso, el instructor del curso, el tema del curso y la url.

Para crear la primera tabla de la base de datos "unicatolica", utiliza el siguiente comando:

```
mysql> CREATE TABLE programas (
  programa_id int NOT NULL,
  programa_name varchar(250) NOT NULL,
  instructor_name varchar(250) NOT NULL,
  technology varchar(50) NOT NULL,
  topic varchar(50) NOT NULL,
  PRIMARY KEY (programa_id)
);
```

Para ver las tablas creadas:

```
mysql>
show tables;
describe programas;
show fields from programas;
```

Para ver la instruccion usada:
Sql
show CREATE table programas;

Para definir una llave primaria, después de haber creado la tabla, usamos la siguiente linea:

alter table programas add PRIMARY KEY (programa_id);

Auto incremento a una tabla creada:
Sql

ALTER TABLE programas MODIFY programa_id int(5) AUTO_INCREMENT;

Primary key

Es un campo (o conjunto de campos) que identifica de forma única cada registro en una tabla.

Características fundamentales:

- Unicidad: No se permiten valores duplicados.
- No nulos: No puede contener NULL.
- Inmutabilidad: Idealmente, su valor no debería cambiar.

Importancia en el Diseño de Bases de Datos

- Integridad de datos: Evita duplicados y registros ambiguos.
- Rendimiento: Facilita la creación de índices clustered (InnoDB) para acelerar consultas.
- Relaciones: Base para vincular tablas mediante claves foráneas.

Sintaxis Básica
Sql

```
-- Creación de tabla con clave primaria
CREATE TABLE usuarios (
   id INT AUTO_INCREMENT PRIMARY KEY,
   nombre VARCHAR(50) NOT NULL,
   email VARCHAR(100) UNIQUE
);
```

Claves Primarias Compuestas
Sql

```
CREATE TABLE pedidos_detalle (
   pedido_id INT,
   producto_id INT,
   cantidad INT,
   PRIMARY KEY (pedido_id, producto_id) -- Clave primaria compuesta
);
```

Agregar Clave Primaria a una Tabla Existente
Sql

```
ALTER TABLE empleados
ADD PRIMARY KEY (dni);
```

Eliminar una Clave Primaria
Sql

```
ALTER TABLE empleados
DROP PRIMARY KEY;
```

Auto Incremento
Sql

```
CREATE TABLE productos (
   id INT PRIMARY KEY AUTO_INCREMENT, -- Valor único automático
   nombre VARCHAR(100)
);
```

Auto incremento a una tabla creada:
Sql

```
ALTER TABLE empleados MODIFY empleado_id int(5) AUTO_INCREMENT;
```

Consejos

- Selección del Campo Ideal:
  - Usar campos numéricos (ej: INT, BIGINT) para optimizar rendimiento.
- Evitar campos variables como VARCHAR a menos que sea necesario (ej: códigos únicos).
- Índices Clustered: En InnoDB, la clave primaria define el orden físico de los datos. Optimizar para consultas frecuentes.
- Evitar Sensibilidad: No usar datos sensibles (ej: números de tarjeta de crédito) como clave primaria.

Claves Naturales vs. Surrogadas:

- Naturales: Usar datos existentes (ej: DNI). Riesgo: Pueden cambiar.
- Surrogadas: Valores generados (ej: AUTO_INCREMENT). Recomendadas para estabilidad.

Foreing Key

Las claves foráneas son un mecanismo fundamental para garantizar la integridad referencial entre tablas.

Características Clave

- Integridad Referencial: Asegura que los valores en la clave foránea existan en la tabla referenciada.
- Acciones en Cascada (CASCADE):
  - ON DELETE CASCADE: Elimina registros dependientes si se borra el padre.
  - ON UPDATE CASCADE: Actualiza automáticamente los valores dependientes.

Sintaxis Básica

Sql

```
-- Creación de tabla con clave foránea
CREATE TABLE tabla_hija (
   id INT PRIMARY KEY,
   padre_id INT,
   FOREIGN KEY (padre_id)
      REFERENCES tabla_padre(id)
      ON DELETE CASCADE
      ON UPDATE NO ACTION
);
```

Agregar Clave Foránea a Tabla Existente usando ADD CONSTRAINT (AGREGAR RESTRICCIÓN)

Sql

```
ALTER TABLE tabla_hija
ADD CONSTRAINT fk_padre
FOREIGN KEY (padre_id)
REFERENCES tabla_padre(id);
```

Visualizar

Sql

```
SHOW tabla_hija;
```

ó

```
DESCRIBE tabla_hija;
```

ó

```
SHOW CREATE TABLE tabla_hija;
```

Eliminar una Clave Foránea

Sql
ALTER TABLE tabla_hija
DROP FOREIGN KEY fk_padre;

Consejos

- Nomenclatura Clara: Usar prefijos como fk_ para nombrar restricciones (ej: fk_pedidos_clientes).
- Índices Automáticos: MySQL indexa automáticamente las claves foráneas, pero verificar su rendimiento con EXPLAIN.
- Evitar Circularidad: No crear relaciones circulares (ej: Tabla A → Tabla B → Tabla A).
- Motor InnoDB: Solo InnoDB soporta claves foráneas en MySQL. Asegurar su uso:

Sql
ENGINE = InnoDB;

- Monitorear Cascadas: Las acciones en cascada pueden afectar el rendimiento en tablas grandes.

Consultas SQL básicas

Ya tienes tu primera base de datos y tu primera tabla. Ahora es el momento de añadir algunos datos a la tabla.

Vamos a añadir algunas filas con algunos cursos populares disponibles en nuestro catálogo con la sentencia INSERT.

Sql

***INSERT** INTO*
programas (programa_id, programa_name, instructor_name, technology, topic) VALUES
(1, 'Introduction to SQL', 'Izzy Weber', 'SQL','Data Manipulation'),
(2, 'Database Design', 'Lis Sulmont', 'SQL', 'Data Engineering'),
(3, 'Data Manipulation with pandas', 'Richie Cotton', 'R','Programming'),
(4, 'Generative AI Concepts', 'Daniel Tedesco', 'Theory', 'AI & Machine Learning');

Para comprobar que las filas se han añadido correctamente, vamos a ver todas las filas de la tabla con la sentencia SELECT:

***SELECT** * FROM programa;*

```
+-----------+------------------------------   +-----------------   +------------   +----------------------   +
| programa_id | programa_name    | instructor_name    | technology   | topic                |
+-----------+------------------------------   +-----------------   +------------   +----------------------   +
|       1 | Introduction to SQL          | Izzy Weber         | SQL          | Data Manipulation  |
|       2 | Database Design              | Lis Sulmont        | SQL          | Data Engineering   |
|       3 | Data Manipulation with pandas | Richie Cotton     | R            | Programming        |
|       4 | Generative AI Concepts       | Daniel Tedesco     | Theory       | AI & Machine Learning |
+------- +------------------------------   +-----------------   +------------   +----------------------   +
```

4 rows in set (0,00 sec)

Hay un error en los datos que acabamos de añadir. La tecnología del programa Manipulación de Datos con pandas es un Python, ni R. Vamos a solucionar el error con la sentencia **UPDATE** junto con la cláusula WHERE para especificar la fila a actualizar

Sql

UPDATE programas SET technology = 'Python' WHERE programa_id = 3;

*SELECT * FROM programa;*

```
+-----------+------------------------------   +-----------------   +------------   +----------------------   +
| programa_id | programa_name    | instructor_name    | technology   | topic                |
+-----------+------------------------------   +-----------------   +------------   +----------------------   +
|       1 | Introduction to SQL          | Izzy Weber         | SQL          | Data Manipulation  |
|       2 | Database Design              | Lis Sulmont        | SQL          | Data Engineering   |
|       3 | Data Manipulation with pandas | Richie Cotton     | Python       | Programming        |
|       4 | Generative AI Concepts       | Daniel Tedesco     | Theory       | AI & Machine Learning |
+-----------+------------------------------   +-----------------   +------------   +----------------------   +
```

4 rows in set (0,00 sec)

Por último, puede que quieras eliminar uno de los registros de tu tabla. Por ejemplo, eliminemos el programa Conceptos de IA Generativa:

***DELETE** from programas WHERE programa_name = 'Generative AI Concepts';*

*SELECT * FROM programas;*

```
+-----------      +------------------------------      +-----------------      +------------      +----------------  +
| programa_id | programa_name                | instructor_name     | technology   | topic          |
+-----------      +------------------------------      +-----------------      +------------      +----------------  +
|       1          | Introduction to SQL          | Izzy Weber          | SQL          | Data Manipulation|
|       2          | Database Design              | Lis Sulmont         | SQL          | Data Engineering  |
|       3          | Data Manipulation with pandas | Richie Cotton      | Python       | Programming       |
+-----------      +------------------------------      +-----------------      +------------      +-------------------+
```

3 rows in set (0,00 sec)

DROP TABLE (Eliminar tablas)

Elimina una tabla de la base de datos.
Sintaxis:

sql
DROP TABLE [IF EXISTS] tabla;

Ejemplo:
sql
DROP TABLE IF EXISTS programas;

Advertencia: Esta acción elimina la tabla y todos sus datos permanentemente.

Indexación

Por defecto, cuando MySQL tiene que encontrar unas filas, empezará por la primera y luego leerá de la cabeza a la cola. Este comportamiento puede ser problemático cuando tus bases de datos tienen millones de filas, ya que el proceso puede ser muy lento.

Una buena forma de acelerar la forma en que MySQL busca filas en tus tablas es creando un índice. Un índice de un componente de tus tablas que MySQL utiliza para recuperar filas con columnas específicas más rápidamente.

la indexación es una estructura de datos que permite acelerar la búsqueda y recuperación de datos en una tabla. Funciona como un índice de un libro, que permite ir directamente a la página deseada sin tener que leer todas las páginas. Los índices se crean en una o más columnas de una tabla y contienen los valores de esas columnas, ordenados de forma que MySQL pueda encontrar rápidamente los datos que coincidan con una consulta.

¿Cómo funciona la indexación?

- **Identificación rápida de datos:**

Cuando se ejecuta una consulta que busca datos en una tabla, MySQL utiliza los índices para identificar rápidamente las filas que coinciden con los criterios de búsqueda.

- **Aceleración de la búsqueda:**
  Los índices permiten que MySQL evite tener que leer cada fila de la tabla para encontrar los datos deseados, lo que acelera significativamente el proceso de búsqueda.

- **Mejora del rendimiento:**
  La indexación contribuye a mejorar el rendimiento general de las consultas, especialmente en tablas grandes con muchos datos.

Vamos a crear un índice en nuestra tabla de cursos llamado "idx_curso" sobre la columna "course_id".

Sql
CREATE INDEX idx_programa
ON programa (programa_id);

Hasta ahora, sólo hemos trabajado con la tabla de cursos. Pero sólo empiezas a aprovechar todo el potencial de las bases de datos relacionales, como MySQL, cuando trabajas con varias tablas a la vez.

Join

La herramienta mágica para combinar varias tablas es la operación JOIN. Imagina que tenemos una segunda tabla en nuestra base de datos llamada "instructores" que contiene información básica sobre los instructores de los programas Unicatolica. El siguiente script crea la tabla y añade algunas filas (ficticias):

```
mysql> CREATE TABLE instructors (
  instructor_id int NOT NULL,
  instructor_name varchar(250) NOT NULL,
  role varchar(500) NOT NULL,
  number_courses int NOT NULL,
  PRIMARY KEY (instructor_id)
);


INSERT INTO
    instructors (instructor_id, instructor_name, role, number_courses) VALUES
    (1, 'Lis Sulmont', 'Data Scientist', 4),
    (2, 'Daniel Tedesco', 'Business Analyst', 1),
    (3, 'Richie Cotton', 'Data Evangelist',5),
    (4, 'Izzy Weber', 'Data Engineer', 2),
    (5, 'James Chapman', 'Data Analyst',3);
```

Imagina que quieres fusionar las dos tablas para obtener la información de los cursos, así como los datos asociados al instructor. Utilizaremos un INNER JOIN para obtener sólo la información de los cursos que aparecen en la tabla de cursos. La columna común para hacer la unión es "nombre_instructor".

Sql
*SELECT * FROM programas*
INNER JOIN instructors ON programas.instructor_name = instructors.instructor_name;

ó

Sql
*SELECT * FROM programas*
JOIN instructors ON programas.instructor_name = instructors.instructor_name;

```
+-----------+---------------------------------+-----------------+------------+--------------------+---------------+-----------------+----------------+----------------+
| course_id | course_name                     | instructor_name | technology | topic              | instructor_id | instructor_name | role           | number_courses |
+-----------+---------------------------------+-----------------+------------+--------------------+---------------+-----------------+----------------+----------------+
|         2 | Database Design                 | Lis Sulmont     | SQL        | Data Engineering   |             1 | Lis Sulmont     | Data Scientist |              4 |
|         3 | Data Manipulation with pandas | Richie Cotton   | Python     | Programming        |             3 | Richie Cotton   | Data Evangelist |              5 |
|         1 | Introduction to SQL             | Izzy Weber      | SQL        | Data Manipulation  |             4 | Izzy Weber      | Data Engineer  |              2 |
+-----------+---------------------------------+-----------------+------------+--------------------+---------------+-----------------+----------------+----------------+
3 rows in set (0,00 sec)
```

Ver mas en :

https://redtauros.com/Clases/Bases_Datos/05_Normalizacion_2_Ejemplo.pdf

https://programacionymas.com/blog/como-funciona-inner-left-right-full-join

Cambios o Modificaciones (RENAME, CHANGE Y MODIFY)

Cambiar o renombrar el nombre de la tabla *programas* a *programs*
Sql
ALTER TABLE programas RENAME programs;
Show tables;

Adicionar columnas a una tabla llamada programs
Sql
ALTER TABLE programs ADD COLUMN IF NOT EXISTS nuevo_nombre_columna tipo_de_datos;

Adicionar varias columnas a una tabla llamada programs
Sql
ALTER TABLE programs
ADD COLUMN IF NOT EXISTS nuevo_nombre_columna tipo_de_datos,
ADD COLUMN IF NOT EXISTS nuevo_nombre_columna2 tipo_de_datos2

Change y Modify
Cambiar el nombre de la columna *instructor_name* a *nombre_docente* con la definición del tipo de datos VARCHAR(100), de la tabla *programs.*
Sql
ALTER TABLE programs CHANGE instructor_name nombre_docente VARCHAR(100);
DESCRIBE programs;

Cambiar solo el tipo de dato de la columna *nombre_docente* VARCHAR(150), conservando el mismo nombre, de la tabla *programs.*
Sql
ALTER TABLE programs CHANGE nombre_docente nombre_docente VARCHAR(150);

DESCRIBE programs;

Cambiar solo el tipo de dato de la columna *nombre_docente*, conservando el mismo nombre usando modify, de la tabla *programs.*
Sql
ALTER TABLE *programs* MODIFY *nombre_docente CHAR(200);*
DESCRIBE programs;

Cambiar solo el tipo de dato de la columna *nombre_docente*, especificando que puede ser nulo, de la tabla *programs.*
Sql
ALTER TABLE programs MODIFY nombre_docente CHAR(200) NULL;
DESCRIBE programs;

Cambiar solo el tipo de datos *VARCHAR(100)* de la columna *nombre_docente*, conservando el mismo nombre, de la tabla *programs.*
Sql
ALTER TABLE programs MODIFY nombre_docente VARCHAR(100);
DESCRIBE programs;

Update
Cambiar todos los valores de la columna valor_2 por en numero 2 en la tabla programs
Sql
update programs SET valor_2 = 2;

Vaciar los datos de una tabla llamada programs.
DELETE FROM Clientes;

Borrar
Borrar columna de una tabla llamada programs, el campo docentes.
*Sq*l
alter table programs drop docentes;

Borrar una tabla
Sql
DROP TABLE nombre_de_la_tabla;

Vaciar una tabla
Sql
TRUNCATE TABLE nombre_de_la_tabla;

ó
Sql
DELETE FROM nombre_de_la_tabla;

Buenas prácticas de MySQL

Medidas de seguridad

Las bases de datos modernas pueden almacenar tablas con millones de filas. Con tantos datos y aplicaciones basados en ellos, mantenerlos seguros es vital.

Las bases de datos están sujetas a riesgos comunes, como la mala gestión del acceso a la base de datos, las contraseñas débiles y las inyecciones SQL. Para hacer frente a estas amenazas y debilidades, se pueden aplicar una serie de estrategias y medidas, como configurar los privilegios de acceso en función del usuario, crear desencadenadores en los servidores SQL e implantar metodologías de encriptación. Un buen lugar para iniciarse en la seguridad SQL es nuestro Curso de Diseño de Bases de Datos.

Optimización del rendimiento

A medida que avances en tu viaje de codificación SQL, empezarás a darte cuenta de la importancia del rendimiento. Una consulta SQL puede escribirse de distintas formas. Todas funcionan, pero algunas son más eficaces que otras.

Hay muchas estrategias y trucos que pueden ayudarte a mejorar el rendimiento de tu código SQL, incluidas las subconsultas y las cláusulas SQL, como WHERE, HAVING y DISTINCT.

Pero la eficacia no es sólo una cuestión técnica, sino también de orden y legibilidad. Con pequeños cambios en tu forma de escribir SQL, como el uso de alias y comentarios, puedes marcar una gran diferencia que te ayudará a ti y al resto de tu equipo.

Curso

https://www.youtube.com/playlist?list=PLUaB-1hjhk8FE_XZ87vPPSfHqb6OcM0cF

Practica

https://sh007.webhostbox.net:2083/cpsess2543513392/3rdparty/phpMyAdmin/
Username aicon4f4

Ejercicio

Realizar el análisis completo del diseño de una base de datos para una agenda, el cual tiene esta información:

- Nombre de la empresa y tipo de sociedad (sas, ltda, anónima, por acciones..)
- Tipo de negocio (Tecnología, Importador, Comercio de ropa, etc)
- Dirección, Ciudad, departamento y país de la sede principal de la empresa
- Nombres, Apellidos, Celular, email, Whatsapp y/o Fijo, Cargo (propietario, gerente, ceo, comercial..) del contacto
- Nombres, Apellidos, , Cargo, Celular, email de la persona que atiende el contacto de nuestra empresa.

Realizar :

1. Los tres primeras Formas Normales de la normalización.
2. Los modelos Entidad Relación, Lógico y Físico
3. Crear las instrucciones MySQL
4. Crear las tablas en el PHP MyAdmin

Timestamp vs Datetime

```
mysql> show variables like '%time_zone%';
+------------------+--------------------+
| Variable_name    | Value              |
+------------------+--------------------+
| system_time_zone | India Standard Time |
| time_zone        | Asia/Calcutta      |
+------------------+--------------------+

mysql> create table datedemo(
    -> mydatetime datetime,
    -> mytimestamp timestamp
    -> );

mysql> insert into datedemo values ((now()),(now()));

mysql> select * from datedemo;
+---------------------+---------------------+
| mydatetime          | mytimestamp         |
+---------------------+---------------------+
| 2011-08-21 14:11:09 | 2011-08-21 14:11:09 |
+---------------------+---------------------+

mysql> set time_zone="america/new_york";

mysql> select * from datedemo;
+---------------------+---------------------+
| mydatetime          | mytimestamp         |
```

+---------------------+---------------------+

| 2011-08-21 14:11:09 | 2011-08-21 04:41:09 |

+---------------------+---------------------+

Fuentes

https://www.datacamp.com/es/tutorial/my-sql-tutorial

https://quickref.me/mysql

https://www.w3schools.com/sql/sql_intro.asp «««« Leerlo!!!!

https://dev.mysql.com/doc/

https://platzi.com/tutoriales/50-sql-mysql-2016/1564-que-es-ddl-dml-dcl-y-tcl-integridad-referencial/

https://dev.mysql.com/doc/refman/8.4/en/create-table-foreign-keys.html

Plataformas interactivas: SQLZoo, LeetCode, HackerRank.