[Teoría] MySQL

Sitio:	Centros - Cádiz	Imprimido por:	Tirado Ramos, Adrián
Curso:	Bases de Datos	Día:	martes, 16 de abril de 2024, 12:32
Libro:	[Teoría] MySQL		

Descripción

Tabla de contenidos

1. Introducción MySQL

Introducción al Temario de Base de Datos sobre MySQL

En este apartado del temario, nos adentraremos en el fascinante mundo de los Sistemas de Gestión de Bases de Datos (SGBD) con un enfoque específico en MySQL. Un SGBD es una herramienta esencial en el ámbito de la informática, encargada de organizar y gestionar eficientemente grandes cantidades de datos. MySQL, por su parte, se destaca como uno de los SGBD más populares y ampliamente utilizados en todo el mundo.

¿Qué es MySQL?

MySQL es un sistema de gestión de bases de datos relacional, de código abierto y altamente confiable. Desde su creación, ha evolucionado continuamente, adaptándose a las crecientes demandas de la industria y manteniendo su posición como una opción predilecta tanto para pequeños proyectos como para aplicaciones empresariales a gran escala.

Objetivo del Temario

El propósito principal de este temario es proporcionar a los estudiantes una comprensión profunda de los fundamentos teóricos y prácticos relacionados con MySQL. Desde la instalación y configuración hasta la optimización de consultas y la implementación de medidas de seguridad, exploraremos cada aspecto crucial que constituye el dominio de la administración de bases de datos con MySQL.

Contenido del Temario

A lo largo de las diversas secciones, abordaremos temas que van desde la estructura básica de una base de datos hasta las complejidades de las transacciones y la integración con aplicaciones. Además, exploraremos casos prácticos y ejemplos concretos para aplicar los conocimientos adquiridos de manera efectiva.

Preparación para el Mundo Profesional

Este temario está diseñado para equipar a los estudiantes con las habilidades necesarias para afrontar desafíos del mundo real. La comprensión profunda de MySQL les permitirá gestionar datos de manera eficiente, garantizando la integridad y la disponibilidad de la información, aspectos cruciales en el desarrollo de aplicaciones y sistemas informáticos modernos.

En resumen, este viaje por el universo de MySQL proporcionará a los estudiantes las herramientas teóricas y prácticas necesarias para convertirse en profesionales competentes en el campo de la gestión de bases de datos. ¡Comencemos este emocionante recorrido hacia el dominio de MySQL y su aplicación en el vasto mundo de la informática!

2. MySQL: El Cliente de MySQL-Server

1.1 Introducción al Cliente MySQL

MySQL no es solo un servidor de bases de datos, sino también un conjunto de herramientas que permiten interactuar con él de manera efectiva. En este primer capítulo, nos centraremos en el cliente de MySQL, una interfaz de línea de comandos que facilita la administración y manipulación de datos en el servidor.

1.2 Acceso al Cliente MySQL

Para acceder al cliente MySQL desde la línea de comandos, se utiliza el siguiente formato:

```
mysql [opciones] [baseDeDatos]
```

- `[opciones]`: Parámetros adicionales que configuran el comportamiento del cliente.
- `[baseDeDatos]`: Nombre de la base de datos a la que se desea acceder.

1.3 Opciones Comunes del Cliente MySQL

- `-help`: Muestra un resumen de las opciones disponibles.

- `-h` o `--host`: Especifica el host del servidor MySQL.
- `-u` o `--user`: Indica el nombre de usuario para la conexión.
- `-p` o `--password`: Solicita la contraseña durante el inicio de sesión.
- `-P` o `--port`: Especifica el puerto en el que el servidor MySQL está escuchando.

La opción más estándar y que usarás la mayoría de las veces será:

```
mysql -u root -p
```

Para después introducir el password de acceso.

1.4 Ejemplos Prácticos

Ejemplo 1: Iniciar sesión en el servidor MySQL en localhost con el usuario 'usuario' y solicitar la contraseña:

```
mysql -h localhost -u usuario -p
```

Ejemplo 2: Conectar al servidor MySQL en un host remoto en el puerto 3306 y seleccionar la base de datos 'ventas':

```
mysql -h servidor_remoto -u usuario_remoto -p -P 3306 ventas
```

1.5 Interacción Básica con el Cliente MySQL

Una vez conectado, el cliente MySQL permite la ejecución de comandos SQL directamente desde la línea de comandos. Algunos comandos básicos incluyen:

- `SHOW DATABASES;`: Muestra las bases de datos disponibles.
- `USE nombreDeBaseDeDatos;`: Selecciona una base de datos específica.
- `SHOW TABLES;`: Muestra las tablas en la base de datos actual.

1.6 Desconexión del Cliente MySQL

Para desconectar del cliente MySQL, simplemente utilizamos el siguiente comando:

```
QUIT
```

o simplemente presionamos `Ctrl + D`.

Este capítulo sienta las bases para el uso del cliente MySQL, proporcionando a los estudiantes las herramientas necesarias para interactuar con el servidor y realizar operaciones fundamentales en la gestión de bases de datos. En los capítulos siguientes, exploraremos en detalle las operaciones más avanzadas y potentes que el cliente MySQL ofrece. ¡Sigamos explorando el emocionante mundo de MySQL!

2.1. Ejercicios

Ejercicio 1:

Conéctate al servidor MySQL local con el usuario "root" y selecciona la base de datos "information_schema". No necesitas ingresar contraseña.

Ejercicio 2:

Utiliza el cliente MySQL para mostrar todas las bases de datos disponibles en el servidor al que estás conectado.

Ejercicio 3:

Muestra las tablas correspondientes a la BBDD 'information_schema'.

3. Ejecución de consultas

En este capítulo, exploraremos la ejecución de consultas en MySQL, centrándonos en el comando `SELECT`. Veremos ejemplos prácticos utilizando funciones esenciales y abordaremos la diferencia entre escribir comandos en mayúsculas y minúsculas.

3.1 Comando SELECT y Funciones Básicas

El comando SELECT se utiliza para recuperar datos de una o varias tablas en MySQL. Empecemos con dos funciones comunes:

```
mysql> SELECT VERSION(), CURRENT_DATE();

+-------------------------+----------------+
| version()               | current_date() |
+-------------------------+----------------+
| 8.0.36-0ubuntu0.23.10.1 | 2024-02-05     |
+-------------------------+----------------+
1 row in set (0,00 sec)
```

Este ejemplo ilustra cómo obtener información del sistema y datos actuales en una sola consulta.

```
mysql> select user();select now();

+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0,00 sec)

+---------------------+
| now()               |
+---------------------+
| 2024-02-05 12:59:13 |
+---------------------+
1 row in set (0,00 sec)
```

La consulta `mysql> select user();` y `select now();` son dos consultas SQL que se están ejecutando de forma secuencial en el cliente MySQL.

La primera consulta, `select user();`, devuelve información sobre el usuario actual que está conectado al servidor MySQL, incluyendo el nombre de usuario y el host desde el cual se ha establecido la conexión.

La segunda consulta, `select now();`, devuelve la fecha y la hora actuales del sistema del servidor MySQL.

Además es interesante saber que NO es necesario escribir un comando en una sola línea, así que los comandos que requieran de varias líneas no son un problema. MySQL determinará dónde finaliza la sentencia cuando encuentre el punto y coma, no cuando encuentre el fin de línea:

```
mysql> select user(),
    -> current_date();
+----------------+----------------+
| user()         | current_date() |
+----------------+----------------+
| root@localhost | 2024-02-05     |
+----------------+----------------+
1 row in set (0,00 sec)
```

El prompt se comunica con el usuario dándole información sobre qué ocurre, por ejemplo:

```
-> Espera la siguiente línea del comando
'> Espera que se cierre una comilla simple
"> Espera que se cierre una comilla doble
mysql> Listo para una nueva consulta.
```

Otra forma de ejecutar comandos SQL es almacenarlos en un fichero de texto y mandarlo a ejecución mediante el comando source, que recibe como parámetro un fichero de comandos y ejecuta uno por uno todos y cada uno de ellos.

#ejecución del script de creación crear_bbdd_dragonBall.sql

```
mysql> source /home/teo/crear_bdd_dragonBall.sql
```

También es posible ejecutar los comandos de un fichero de texto desde la shell, esto es, en modo batch, redirigiendo al cliente mysql un fichero de entrada. Esto es útil para tareas administrativas donde se ejecutan varios ficheros de comandos, además de otras tareas de mantenimiento del servidor a través de un shell script:

#ejecución en modo batch

```
mysql -u root -pPassUsuario <crear_bbdd_dragonBall.sql
```

#ejecución en modo batch almacenando resultados

```
mysql -u root -pPassUsuario <crear_bbdd_dragonBall.sql >resultado.txt
```

3.2 Diferencia entre Mayúsculas y Minúsculas

MySQL es en gran medida insensible a mayúsculas y minúsculas en cuanto a los comandos SQL. Sin embargo, los nombres de las bases de datos y las tablas sí son sensibles a mayúsculas y minúsculas dependiendo de la configuración del sistema. Por lo tanto, `SELECT` y `select` son equivalentes, pero `BaseDatos` y `basedatos` pueden referirse a entidades diferentes.

Es importante señalar que, aunque MySQL es insensible a mayúsculas y minúsculas en muchos casos, es una buena práctica escribir palabras clave (como `SELECT`, `FROM`, etc.) en mayúsculas para mejorar la legibilidad del código.

3.1. Ejercicios

Ejercicio.
Abre y estudia lo que ocurre línea a línea en el archivo 'crear_bbdd_dragonBall.sql', después utilizalo para cargar la base de datos, y después:
Lista las tablas.
Lista el interior de las tablas.

Descarga el archivo aquí:

4. Lenguaje de Definición de Datos (DDL) en MySQL

El Lenguaje de Definición de Datos (DDL) en MySQL se utiliza para definir, modificar y eliminar estructuras de base de datos, como tablas, índices y restricciones. Las tres instrucciones principales del DDL son CREATE, DROP y ALTER, cada una con su función específica en la administración de la base de datos.

1. Instrucción CREATE

La instrucción CREATE se utiliza para crear nuevos objetos en la base de datos, como tablas, índices, vistas o procedimientos almacenados. Permite definir la estructura y las características de los objetos que se van a crear.

Ejemplo:

```
-- Crear una tabla para almacenar información sobre los personajes

CREATE TABLE Personajes (

    id_personaje INT PRIMARY KEY,

    nombre VARCHAR(255),

    raza VARCHAR(255),

    nivel_poder INT

);
```

En este ejemplo, estamos utilizando la instrucción CREATE TABLE para crear una nueva tabla llamada Personajes, con columnas para el identificador del personaje, el nombre, la raza y el nivel de poder.

2. Instrucción DROP

La instrucción DROP se utiliza para eliminar objetos existentes de la base de datos, como tablas, índices o vistas. Elimina completamente el objeto y todos los datos asociados a él.

Ejemplo

```
-- Eliminar la tabla de técnicas

DROP TABLE Tecnicas;
```

En este ejemplo, estamos utilizando la instrucción DROP TABLE para eliminar la tabla Tecnicas de la base de datos, junto con todas las técnicas asociadas a ella.

3. Instrucción ALTER

La instrucción ALTER se utiliza para modificar la estructura de un objeto existente en la base de datos, como agregar, modificar o eliminar columnas de una tabla, cambiar el nombre de un objeto o modificar las restricciones.

Ejemplo

```
-- Agregar una nueva columna a la tabla Personajes

ALTER TABLE Personajes ADD COLUMN universo VARCHAR(255);
```

En este ejemplo, estamos utilizando la instrucción ALTER TABLE para agregar una nueva columna llamada universo a la tabla Personajes, que indica el universo al que pertenece el personaje.

Estas instrucciones del Lenguaje de Definición de Datos (DDL) son fundamentales para la administración y el diseño de bases de datos en MySQL, permitiendo crear, eliminar y modificar la estructura de los objetos según sea necesario.

5. Creación de bases de datos en MySQL

La creación de bases de datos en MySQL es una parte fundamental de la administración de bases de datos. MySQL proporciona varias instrucciones que permiten crear, mostrar y seleccionar bases de datos según sea necesario. Aquí explicaremos las instrucciones CREATE DATABASE, SHOW DATABASES y USE.

1. Instrucción CREATE DATABASE

La instrucción `CREATE DATABASE` se utiliza para crear una nueva base de datos en MySQL. Proporciona opciones adicionales para controlar el proceso de creación, como la posibilidad de especificar si la base de datos ya existe y si deseamos utilizar un nombre de esquema en lugar de una base de datos.

Sintaxis:

```
CREATE DATABASE [IF NOT EXISTS] nombre_de_la_base_de_datos
    [CHARACTER SET nombre_del_juego_de_caracteres]
    [COLLATE nombre_del_juego_de_caracteres]
```

Parámetros:

- **IF NOT EXISTS (OPCIONAL):** Esta cláusula evita que se produzca un error si la base de datos ya existe. Si se especifica esta opción y la base de datos ya existe, MySQL no realizará ninguna acción y no mostrará ningún mensaje de error.
- **nombre_de_la_base_de_datos:** Es el nombre de la base de datos que se va a crear.
- **CHARACTER SET (OPCIONAL):** Esta opción permite especificar el juego de caracteres que se utilizará para la base de datos. Define cómo se almacenan y manejan los datos de texto dentro de la base de datos.
- **COLLATE (OPCIONAL):** Esta opción permite especificar la regla de comparación que se utilizará para comparar los valores de texto dentro de la base de datos. Define cómo se ordenan y comparan los datos de texto.

Ejemplo:

```
CREATE DATABASE IF NOT EXISTS dragon_ball
    CHARACTER SET Latin1
    COLLATE latin1_spanish_ci;
```

En este ejemplo, estamos creando una nueva base de datos llamada `dragon_ball`. Utilizamos la cláusula `IF NOT EXISTS` para evitar errores si la base de datos ya existe. También especificamos el juego de caracteres y la regla de comparación para la base de datos, utilizando `Latin1` como juego de caracteres y `latin1_spanish_ci` como regla de comparación.

Es interesante tener en cuenta el juego de caracteres si se quieren utilizar tildes en los campos (no recomendable). Con respecto a colación (regla de comparación) especifica cómo tratar el alfabeto del juego de caracteres, es decir, cómo se ordena (por ejemplo, la ñ después de la m y n y no conforme a la tabla de códigos ascii), y cómo se comparan los caracteres (por ejemplo, si Á es igual a á).

La instrucción `CREATE DATABASE` es esencial para la creación de nuevas bases de datos en MySQL. Proporciona opciones adicionales para controlar el proceso de creación y definir las configuraciones de caracteres y reglas de comparación para la base de datos.

2. Instrucción SHOW DATABASES

La instrucción `SHOW DATABASES` se utiliza para mostrar una lista de todas las bases de datos disponibles en el servidor MySQL al que estás conectado.

Sintaxis:

```
SHOW DATABASES;
```

Este comando mostrará una lista de todas las bases de datos disponibles en el servidor MySQL.

3. Instrucción USE

La instrucción `USE` se utiliza para seleccionar una base de datos específica en la que se trabajarán las consultas siguientes. Una vez seleccionada una base de datos, todas las operaciones de consulta y manipulación de datos se aplicarán a esa base de datos.

Sintaxis:

```
USE nombre_de_la_base_de_datos;
```

Ejemplo:

```
USE dragon_ball;
```

En este ejemplo, se selecciona la base de datos `dragon_ball` para realizar operaciones en ella.

Estas instrucciones son esenciales para crear, administrar y trabajar con bases de datos en MySQL. La instrucción `CREATE DATABASE` se utiliza para crear nuevas bases de datos, `SHOW DATABASES` para mostrar una lista de bases de datos disponibles y `USE` para seleccionar una base de datos específica para trabajar. Con estas herramientas, puedes comenzar a diseñar y administrar tus bases de datos en MySQL de manera efectiva.

6. Modificación de bases de datos en MySQL

La modificación de una base de datos en MySQL implica realizar cambios en las configuraciones y propiedades de la base de datos en sí misma, como el juego de caracteres, la regla de comparación, entre otros aspectos. A continuación, se detallan los aspectos clave de la modificación de una base de datos en MySQL:

1. Instrucción `ALTER DATABASE`

La instrucción `ALTER DATABASE` se utiliza para realizar cambios en las configuraciones y propiedades de una base de datos existente en MySQL. Esta instrucción permite modificar el juego de caracteres, la regla de comparación y otras opciones de la base de datos.

Sintaxis:

```
ALTER DATABASE nombre_de_la_base_de_datos
    [CHARACTER SET nombre_del_juego_de_caracteres]
    [COLLATE nombre_de_la_regla_de_comparacion];
```

2. Ejemplos.

A continuación se presentan ejemplos de modificación de una base de datos relacionada con Dragon Ball:

Ejemplo 1: Modificar el juego de caracteres de la base de datos

```
ALTER DATABASE DragonBall
    CHARACTER SET utf8mb4;
```

En este ejemplo, modificamos el juego de caracteres de la base de datos `DragonBall` para utilizar `utf8mb4`, lo que permite almacenar una amplia gama de caracteres Unicode, adecuado para nombres y descripciones en varios idiomas, incluido el japonés utilizado en Dragon Ball.

Ejemplo 2: Modificar la regla de comparación de la base de datos

```
ALTER DATABASE DragonBall
    COLLATE utf8mb4_unicode_ci;
```

En este ejemplo, modificamos la regla de comparación de la base de datos `DragonBall` para utilizar `utf8mb4_unicode_ci`, que es una regla de comparación sensible a mayúsculas y minúsculas adecuada para la comparación de texto en varios idiomas.

3. Consideraciones Adicionales

- Al modificar una base de datos en MySQL, es importante tener en cuenta el impacto potencial en las tablas y datos almacenados en la base de datos. Los cambios en el juego de caracteres y la regla de comparación pueden afectar cómo se almacenan y manipulan los datos.
- Es posible utilizar la instrucción `ALTER DATABASE` para realizar otros cambios en la base de datos, como establecer otras opciones de configuración o definir privilegios de acceso específicos.

7. Creación de tablas en MySQL.

En MySQL, la creación de tablas es una parte fundamental del diseño y la administración de bases de datos. Permite definir la estructura de una tabla, incluyendo los nombres de las columnas, los tipos de datos que contienen y las restricciones aplicables. A continuación, se detallan los aspectos clave de la creación de tablas en MySQL:

1. Instrucción `CREATE TABLE`

La instrucción `CREATE TABLE` se utiliza para crear una nueva tabla en una base de datos de MySQL. Esta instrucción define la estructura de la tabla, incluyendo los nombres de las columnas, los tipos de datos de cada columna y cualquier restricción aplicable, como claves primarias o claves foráneas.

Sintaxis:

```
CREATE TABLE nombre_de_la_tabla (
    nombre_columna1 tipo_de_dato [restricciones],
    nombre_columna2 tipo_de_dato [restricciones],
    ...
    [restricciones_adicionales]
);
```

2. Tipos de Datos

MySQL admite una variedad de tipos de datos que pueden utilizarse para definir las columnas de una tabla. Aunque se verán más adelante de forma más completa, como aperitivo debes saber que algunos de los tipos de datos más comunes son:

- **Enteros:** `INT`, `BIGINT`, `SMALLINT`, etc.
- **Decimales:** `DECIMAL`, `FLOAT`, `DOUBLE`, etc.
- **Cadenas de Caracteres:** `VARCHAR`, `CHAR`, `TEXT`, etc.
- **Fechas y Tiempos:** `DATE`, `DATETIME`, `TIMESTAMP`, etc.
- **Valores Booleanos:** `BOOL`, `BOOLEAN`.

3. Restricciones

Las restricciones se utilizan para aplicar reglas adicionales a las columnas de una tabla. Aunque se verán más adelante de forma más completa, como aperitivo debes saber algunas restricciones comunes:

- **Clave Primaria (`PRIMARY KEY`):** Identifica de manera única cada fila de la tabla y garantiza que no haya valores duplicados en la columna especificada como clave primaria.
- **Clave Foránea (`FOREIGN KEY`):** Establece una relación entre dos tablas, especificando que los valores de una columna de una tabla deben coincidir con los valores de otra columna en otra tabla.
- **Restricciones de Unicidad (`UNIQUE`):** Garantiza que no haya valores duplicados en la columna especificada.
- **Restricciones de Comprobación (`CHECK`):** Permite especificar una condición que deben cumplir los valores de una columna.

4. Ejemplos

A continuación se muestra un ejemplo de creación de una tabla simple en MySQL:

Ejemplo 1: Creación de la tabla de Personajes

```
CREATE TABLE Personajes (
    id_personaje INT PRIMARY KEY,
    nombre VARCHAR(100),
    raza VARCHAR(50),
    nivel_poder INT,
    planeta_origen VARCHAR(100)
);
```

En este ejemplo, creamos una tabla llamada `Personajes` que almacena información sobre los personajes de Dragon Ball. La tabla tiene columnas para el ID del personaje, nombre, raza, nivel de poder y planeta de origen.

Ejemplo 2: Creación de la tabla de Técnicas

```
CREATE TABLE Tecnicas (
    id_tecnica INT PRIMARY KEY,
    nombre VARCHAR(100),
    descripcion TEXT,
    tipo VARCHAR(50),
    id_personaje INT,
    FOREIGN KEY (id_personaje) REFERENCES Personajes(id_personaje)
);
```

En este ejemplo, creamos una tabla llamada `Tecnicas` que almacena información sobre las técnicas de combate de los personajes de Dragon Ball. La tabla tiene columnas para el ID de la técnica, nombre, descripción, tipo de técnica y el ID del personaje al que pertenece.

5. Consideraciones Adicionales

- Antes de crear una tabla, es importante planificar la estructura de la misma, incluyendo todas las columnas necesarias y las restricciones aplicables.
- Es posible modificar la estructura de una tabla existente utilizando la instrucción `ALTER TABLE`, que permite agregar, modificar o eliminar columnas, así como aplicar o quitar restricciones.
- Al diseñar una base de datos, es importante normalizar las tablas para evitar la redundancia de datos y garantizar la integridad de los mismos.

8. Consulta de la estructura de una tabla

La consulta de la estructura de una tabla en MySQL se puede realizar de varias formas, pero la más común es utilizando la instrucción `DESCRIBE` seguida del nombre de la tabla. Esta consulta devuelve información detallada sobre la estructura de la tabla, incluyendo los nombres de las columnas, los tipos de datos, las restricciones y otros atributos.

Ejemplo:

Supongamos que queremos conocer la estructura de la tabla `Personajes` en nuestra base de datos de Dragon Ball.

```
DESCRIBE Personajes;
```

Este comando mostrará información detallada sobre la estructura de la tabla `Personajes`, incluyendo el nombre de cada columna, el tipo de datos, la longitud (en el caso de columnas de texto), si es una clave primaria, si permite nulos, entre otros.

Resultado Esperado:

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id_personaje  | INT          | NO   | PRI | NULL    | auto_increment |
| nombre        | VARCHAR(100) | YES  |     | NULL    |                |
| raza          | VARCHAR(50)  | YES  |     | NULL    |                |
| nivel_poder   | INT          | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

En este ejemplo, se muestra la estructura de la tabla `Personajes`. La primera columna (`Field`) indica el nombre de la columna, seguida por el tipo de datos (`Type`), si la columna permite valores nulos (`Null`), si es una clave (`Key`), el valor predeterminado (`Default`) y cualquier información adicional (`Extra`).

La consulta `DESCRIBE` es útil para comprender la estructura de una tabla en una base de datos, lo que facilita la escritura de consultas y la manipulación de los datos almacenados en ella.

Es interesante además conocer que se puede preceder directamente la base de datos concatenándola con un punto, de la siguiente forma:

```
DESCRIBE DragonBall.Personajes;
```

Donde 'DragonBall' sería el nombre de la base de datos y 'Personajes' el de la tabla correspondiente.

9. Modificación de tablas.

En MySQL, la modificación de la estructura de una tabla se puede realizar utilizando la instrucción `ALTER TABLE`. Esta instrucción permite agregar, modificar o eliminar columnas, así como también cambiar las restricciones y los atributos de las columnas existentes.

1. Agregar una Nueva Columna:

Para agregar una nueva columna a una tabla existente, se utiliza la siguiente sintaxis:

```
ALTER TABLE nombre_tabla
ADD nombre_columna tipo_dato [restricciones];
```

Ejemplo:

Supongamos que queremos agregar una columna llamada `planeta_origen` a la tabla `Personajes` para almacenar información sobre el planeta de origen de cada personaje.

```
ALTER TABLE Personajes
ADD planeta_origen VARCHAR(100);
```

Resultado Esperado:

```
DESCRIBE Personajes;
```

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id_personaje  | INT          | NO   | PRI | NULL    | auto_increment |
| nombre        | VARCHAR(100) | YES  |     | NULL    |                |
| raza          | VARCHAR(50)  | YES  |     | NULL    |                |
| nivel_poder   | INT          | YES  |     | NULL    |                |
| planeta_origen| VARCHAR(100) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

En este ejemplo, hemos agregado exitosamente una nueva columna llamada `planeta_origen` a la tabla `Personajes`, la cual tiene un tipo de datos VARCHAR con una longitud máxima de 100 caracteres.

2. Modificar una Columna Existente:

Para modificar una columna existente, se utiliza la siguiente sintaxis:

```
ALTER TABLE nombre_tabla
MODIFY COLUMN nombre_columna nuevo_tipo_dato [nuevas_restricciones];
```

Ejemplo:

Supongamos que queremos modificar la columna `nivel_poder` en la tabla `Personajes` para cambiar su tipo de datos de INT a DECIMAL.

```
ALTER TABLE Personajes
MODIFY COLUMN nivel_poder DECIMAL(10,2);
```

Resultado Esperado:

```
DESCRIBE Personajes;
```

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id_personaje  | INT          | NO   | PRI | NULL    | auto_increment |
| nombre        | VARCHAR(100) | YES  |     | NULL    |                |
| raza          | VARCHAR(50)  | YES  |     | NULL    |                |
| nivel_poder   | DECIMAL(10,2)| YES  |     | NULL    |                |
| planeta_origen| VARCHAR(100) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

En este ejemplo, hemos modificado exitosamente la columna `nivel_poder` en la tabla `Personajes` para cambiar su tipo de datos a DECIMAL con una precisión de 10 dígitos y 2 decimales.

La instrucción `ALTER TABLE` ofrece flexibilidad para realizar cambios en la estructura de una tabla existente en MySQL, lo que permite adaptar la base de datos a medida que cambian los requisitos del sistema o las necesidades de almacenamiento de datos.

3. Cambiar el nombre de una columna.

Para cambiar el nombre de una columna en MySQL, se utiliza la instrucción `ALTER TABLE` junto con la cláusula `CHANGE COLUMN`. Esta cláusula permite cambiar el nombre de una columna y, opcionalmente, especificar un nuevo tipo de datos y restricciones para la columna.

La sintaxis general para cambiar el nombre de una columna es la siguiente:

```
ALTER TABLE nombre_tabla
CHANGE COLUMN nombre_columna_antiguo nombre_columna_nuevo tipo_dato [restricciones];
```

Donde:

- `nombre_tabla` es el nombre de la tabla en la que se encuentra la columna.
- `nombre_columna_antiguo` es el nombre actual de la columna que se quiere cambiar.
- `nombre_columna_nuevo` es el nuevo nombre que se desea asignar a la columna.
- `tipo_dato` es el tipo de datos de la columna.
- `restricciones` son las restricciones opcionales que se pueden aplicar a la columna.

Ejemplo:

Supongamos que queremos cambiar el nombre de la columna `raza` a `especie` en la tabla `Personajes` de nuestra base de datos de Dragon Ball.

```
ALTER TABLE Personajes
CHANGE COLUMN raza especie VARCHAR(50);
```

En este ejemplo, estamos cambiando el nombre de la columna `raza` a `especie` y manteniendo el mismo tipo de datos VARCHAR con una longitud máxima de 50 caracteres.

Resultado Esperado:

```
DESCRIBE Personajes;
```

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id_personaje  | INT          | NO   | PRI | NULL    | auto_increment |
| nombre        | VARCHAR(100) | YES  |     | NULL    |                |
| especie       | VARCHAR(50)  | YES  |     | NULL    |                |
| nivel_poder   | DECIMAL(10,2)| YES  |     | NULL    |                |
| planeta_origen| VARCHAR(100) | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

Como resultado, la columna `raza` ha sido renombrada a `especie` en la tabla `Personajes`, y la estructura de la tabla ha sido actualizada correctamente.

4. DROP para borrar columnas y restricciones.

En MySQL, la instrucción `DROP` se utiliza en la modificación de tablas para eliminar elementos de la estructura de una tabla. Esto puede incluir la eliminación de columnas, índices, restricciones, claves externas u otros objetos asociados a la tabla.

DROP COLUMN

Para eliminar una columna de una tabla existente, se utiliza la instrucción `ALTER TABLE` junto con la cláusula `DROP COLUMN`.

La sintaxis general para eliminar una columna es la siguiente:

```
ALTER TABLE nombre_tabla
DROP COLUMN nombre_columna;
```

Donde:

- `nombre_tabla` es el nombre de la tabla de la cual se eliminará la columna.
- `nombre_columna` es el nombre de la columna que se desea eliminar.

Ejemplo:

Supongamos que queremos eliminar la columna `planeta_origen` de la tabla `Personajes` en nuestra base de datos de Dragon Ball.

```
ALTER TABLE Personajes
DROP COLUMN planeta_origen;
```

Este comando eliminará la columna `planeta_origen` de la tabla `Personajes`.

Resultado Esperado:

Después de ejecutar la instrucción `ALTER TABLE`, se espera que la columna `planeta_origen` sea eliminada de la tabla `Personajes`.

```
DESCRIBE Personajes;
```

```
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id_personaje  | INT          | NO   | PRI | NULL    | auto_increment |
| nombre        | VARCHAR(100) | YES  |     | NULL    |                |
| especie       | VARCHAR(50)  | YES  |     | NULL    |                |
| nivel_poder   | DECIMAL(10,2)| YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
```

Como resultado, la columna `planeta_origen` ha sido eliminada de la tabla `Personajes`, y la estructura de la tabla ha sido actualizada correctamente.

DROP FOREIGN KEY

Para eliminar restricciones de claves foráneas en MySQL, se utiliza la instrucción `ALTER TABLE` junto con la cláusula `DROP FOREIGN KEY`.

La sintaxis general para eliminar una restricción de clave foránea es la siguiente:

```
ALTER TABLE nombre_tabla
DROP FOREIGN KEY nombre_restriccion;
```

Donde:

- `nombre_tabla` es el nombre de la tabla de la cual se eliminará la restricción de clave foránea.
- `nombre_restriccion` es el nombre de la restricción de clave foránea que se desea eliminar.

Ejemplo:

Supongamos que tenemos una tabla llamada `Tecnicas` en nuestra base de datos de Dragon Ball que tiene una restricción de clave foránea llamada `fk_id_personaje` que hace referencia a la columna `id_personaje` en la tabla `Personajes`. Deseamos eliminar esta restricción de clave foránea.

```
ALTER TABLE Tecnicas
DROP FOREIGN KEY fk_id_personaje;
```

Este comando eliminará la restricción de clave foránea `fk_id_personaje` de la tabla `Tecnicas`.

10. Borrado de Tablas

Para eliminar una tabla en MySQL, se utiliza la instrucción `DROP TABLE` seguida del nombre de la tabla que se desea eliminar. Esta operación borrará completamente la tabla y todos sus datos, por lo que se debe tener precaución al ejecutarla, ya que la información no podrá ser recuperada.

DROP TABLE

La sintaxis para eliminar una tabla es la siguiente:

```
DROP TABLE nombre_tabla;
```

Donde `nombre_tabla` es el nombre de la tabla que se desea eliminar.

Ejemplo:

Supongamos que queremos eliminar la tabla `Villanos` de nuestra base de datos de Dragon Ball.

```
DROP TABLE Villanos;
```

Este comando eliminará completamente la tabla `Villanos` y todos sus datos asociados.

Resultado Esperado:

Después de ejecutar la instrucción `DROP TABLE`, la tabla `Villanos` habrá sido eliminada de la base de datos.

```
SHOW TABLES;
```

```
+----------------------+
| Tables_in_DragonBall |
+----------------------+
| Personajes           |
| Tecnicas             |
| Torneos              |
| Razas                |
+----------------------+
```

Como resultado, la tabla `Villanos` ya no estará presente en la lista de tablas de la base de datos `DragonBall`.

11. Renombrado de Tablas

En MySQL, se puede renombrar una tabla utilizando la instrucción `ALTER TABLE` seguida de la cláusula `RENAME TO`. Esto permite cambiar el nombre de una tabla existente sin perder los datos que contiene.

RENAME TABLE

La sintaxis para renombrar una tabla es la siguiente:

```
ALTER TABLE nombre_tabla_antiguo
RENAME TO nombre_tabla_nuevo;
```

Donde `nombre_tabla_antiguo` es el nombre actual de la tabla y `nombre_tabla_nuevo` es el nuevo nombre que se desea asignar a la tabla.

Ejemplo:

Supongamos que queremos renombrar la tabla `Tecnicas` a `Ataques` en nuestra base de datos de Dragon Ball.

```
ALTER TABLE Tecnicas
RENAME TO Ataques;
```

Este comando renombrará la tabla `Tecnicas` a `Ataques`.

Resultado Esperado:

Después de ejecutar la instrucción `ALTER TABLE`, la tabla `Tecnicas` habrá sido renombrada a `Ataques`.

```
SHOW TABLES;
```

```
+----------------------+
| Tables_in_DragonBall |
+----------------------+
| Personajes           |
| Ataques              |
| Torneos              |
| Razas                |
+----------------------+
```

Como resultado, la tabla `Tecnicas` ahora se mostrará como `Ataques` en la lista de tablas de la base de datos `DragonBall`.

11.1. Ejercicios de Creación de Base de Datos.

Ejercicio 0: Creación de una base de datos Dragon Ball

1. Utiliza la instrucción `CREATE DATABASE` para crear una nueva base de datos llamada `DragonBall`.
2. Utiliza la instrucción `USE` para seleccionar la base de datos `DragonBall`.

Ejercicio 1: Creación de una tabla de planetas

1. Utiliza la instrucción `CREATE TABLE` para crear una nueva tabla llamada Planetas, otra llamada Villanos y otra llamada Razas. **Este ejercicio debe de cargarse desde un archivo extensión sql.**

La tabla Planetas debe contener las siguientes columnas:

- `id_planeta` (entero, clave primaria)
- `nombre` (cadena de caracteres)
- `tipo` (cadena de caracteres)
- `habitantes` (entero)

Inserta al menos tres registros de planetas en la tabla utilizando la instrucción `INSERT INTO`. Asegúrate de incluir planetas conocidos de la serie Dragon Ball, como "Namek", "Planeta Tierra", etc.

La tabla Villanos debe contener las siguientes columnas:

- `id_villano` (entero, clave primaria)
- `nombre` (cadena de caracteres)
- `raza` (cadena de caracteres)
- `planeta_origen` (cadena de caracteres)
- `nivel_poder` (entero)
- `descripcion` (cadena de caracteres, opcional)

Inserta al menos tres registros de villanos en la tabla utilizando la instrucción `INSERT INTO`. Asegúrate de incluir villanos conocidos de la serie Dragon Ball, como "Freezer", "Cell", "Majin Buu", etc.

La tabla Razas debe contener las siguientes columnas:

- `id_raza` (entero, clave primaria)
- `nombre` (cadena de caracteres)
- `descripcion` (cadena de caracteres, opcional)
- Inserta al menos tres registros de razas en la tabla utilizando la instrucción `INSERT INTO`. Asegúrate de incluir razas conocidas de la serie Dragon Ball, como "Saiyan", "Namekiano", "Androide", etc.

*** Para rellenar los campos, invéntate los datos o búscalos en Google.*

Ejercicio 2: Eliminación de la tabla de villanos

1. Utiliza la instrucción `DROP TABLE` para eliminar la tabla `Villanos` de la base de datos. Asegúrate de que la tabla exista antes de intentar eliminarla.

Ejercicio 3: Modificación de la tabla de razas

1. Utiliza la instrucción `ALTER TABLE` para agregar una nueva columna llamada `descripcion2` a la tabla `Razas`. La columna `descripcion2` debe ser una cadena de caracteres que permita describir las características de cada raza.

2. Actualiza al menos un registro en la tabla `Razas` para incluir una descripción de la raza correspondiente.

12. Implementación de Restricciones

1. REFERENCES

La restricción `REFERENCES` se utiliza junto con la restricción `FOREIGN KEY` para establecer una relación entre dos tablas, especificando la columna de la tabla actual que hace referencia a la columna de otra tabla.

Ejemplo 1 Sintaxis básica:

```
CREATE TABLE Tecnicas (
    id_tecnica INT PRIMARY KEY,
    nombre VARCHAR(100),
    id_personaje INT REFERENCES Personajes(id_personaje)
);
```

Ejemplo 2 Sintaxis completa:

```
CREATE TABLE Tecnicas (
    id_tecnica INT PRIMARY KEY,
    nombre VARCHAR(100),
    id_personaje INT,
    FOREIGN KEY (id_personaje) REFERENCES Personajes(id_personaje)
);
```

En este ejemplo, la restricción `FOREIGN KEY` en la columna `id_personaje` de la tabla `Tecnicas` establece una relación con la columna `id_personaje` de la tabla `Personajes`, lo que garantiza que cada técnica esté asociada a un personaje existente en la tabla `Personajes`.

Por lo tanto se puede observar que la sintaxis completa de la estricción `REFERENCES` es la siguiente:

```
FOREIGN KEY (columna_local) REFERENCES tabla_principal(columna_referenciada)
```

Donde:

- `columna_local` es la columna en la tabla secundaria que contendrá los valores que hacen referencia a la tabla principal.
- `tabla_principal` es la tabla principal que contiene los valores referenciados.
- `columna_referenciada` es la columna en la tabla principal que se referencia.

2. Acciones en Cascada ON DELETE y ON UPDATE

Además de la restricción `REFERENCES`, también podemos especificar acciones adicionales que MySQL debe realizar cuando ocurren ciertos eventos, como la eliminación o actualización de filas en la tabla principal. Estas acciones se especifican utilizando las cláusulas `ON DELETE` y `ON UPDATE`.

Las cláusulas `ON DELETE` y `ON UPDATE` se utilizan para especificar el comportamiento que debe seguir la base de datos cuando se eliminan o actualizan filas relacionadas en una tabla principal.

Ejemplo:

```
CREATE TABLE Tecnicas (
    id_tecnica INT PRIMARY KEY,
    nombre VARCHAR(100),
    id_personaje INT,
    FOREIGN KEY (id_personaje) REFERENCES Personajes(id_personaje) ON DELETE CASCADE ON UPDATE CASCADE
);
```

En este ejemplo, `ON DELETE CASCADE` especifica que si un personaje se elimina de la tabla `Personajes`, todas las técnicas asociadas a ese personaje se eliminarán automáticamente de la tabla `Tecnicas`. Del mismo modo, `ON UPDATE CASCADE` especifica que si se actualiza el ID de un personaje en la tabla `Personajes`, las técnicas asociadas a ese personaje también se actualizarán automáticamente.

Opciones:

- `CASCADE`: Si una fila principal se elimina o actualiza, las filas relacionadas en la tabla secundaria también se eliminan o actualizan automáticamente.
- `SET NULL`: Si una fila principal se elimina o actualiza, las columnas correspondientes en la tabla secundaria se establecen en `NULL`.
- `NO ACTION`: La operación se rechaza si hay filas relacionadas en la tabla secundaria.
- `RESTRICT`: Similar a `NO ACTION`, pero se utiliza para mayor claridad.

3. CONSTRAINT, UNIQUE y CHECK

3.1. CONSTRAINT

La restricción CONSTRAINT se utiliza para aplicar reglas específicas a las columnas de una tabla en MySQL. Estas reglas pueden ser la definición de una clave primaria, una clave foránea, restricciones de integridad referencial, entre otras.

Ejemplo:

```
CREATE TABLE Personajes (
    id_personaje INT PRIMARY KEY,
    nombre VARCHAR(100),
    raza VARCHAR(50),
    CONSTRAINT pk_personaje_id UNIQUE (id_personaje)
);
```

En este ejemplo, la restricción CONSTRAINT se utiliza para definir una clave primaria en la columna `id_personaje` de la tabla `Personajes`. Además, se establece una restricción UNIQUE en la columna `id_personaje`, lo que garantiza que cada valor en esta columna sea único en la tabla.

3.2. UNIQUE

La restricción UNIQUE se utiliza para asegurar que los valores en una columna específica sean únicos en la tabla. Esto significa que no puede haber valores duplicados en la columna definida como única.

Ejemplo:

```
CREATE TABLE Razas (
    id_raza INT PRIMARY KEY,
    nombre VARCHAR(100) UNIQUE,
    descripcion VARCHAR(255)
);
```

En este ejemplo, se define la restricción UNIQUE en la columna `nombre` de la tabla `Razas`, lo que garantiza que cada nombre de raza sea único en la tabla.

3.3. CHECK

La restricción CHECK se utiliza para aplicar una condición específica a los valores de una columna. Esta condición debe evaluarse como verdadera para que se permita la inserción o actualización de datos en la tabla.

Ejemplo:

```
CREATE TABLE Tecnicas (
    id_tecnica INT PRIMARY KEY,
    nombre VARCHAR(100),
    descripcion VARCHAR(255),
    tipo ENUM('Ataque', 'Defensa', 'Soporte'),
    nivel_poder INT CHECK (nivel_poder >= 0)
);
```

En este ejemplo, se define la restricción CHECK en la columna `nivel_poder` de la tabla `Tecnicas`, asegurando que el valor de `nivel_poder` no sea negativo. Esto garantiza que solo se inserten o actualicen técnicas con un nivel de poder igual o mayor que cero.

Consideraciones Adicionales:

- Las restricciones CONSTRAINT, UNIQUE y CHECK son herramientas poderosas para garantizar la integridad de los datos en una base de datos MySQL.
- Es importante aplicar estas restricciones según las necesidades específicas del diseño de la base de datos y las reglas de negocio del sistema.
- Las restricciones pueden aplicarse durante la creación de la tabla o posteriormente mediante la modificación de la estructura de la tabla con la instrucción ALTER TABLE.

4. PRIMARY KEY y DEFAULT

- `PRIMARY KEY`: Identifica de manera única cada fila en la tabla y garantiza que no haya valores duplicados en la columna especificada.
- `DEFAULT`: Establece un valor predeterminado para una columna cuando no se proporciona un valor en una instrucción `INSERT`.

4.1. PRIMARY KEY

La restricción PRIMARY KEY se utiliza para identificar de manera única cada fila en una tabla y garantizar que no haya valores duplicados en la columna especificada. Una columna con la restricción PRIMARY KEY no puede contener valores nulos y puede haber solo una PRIMARY KEY en una tabla.

Ejemplo:

```
CREATE TABLE Personajes (
    id_personaje INT PRIMARY KEY,
    nombre VARCHAR(100),
    raza VARCHAR(50)
);
```

En este ejemplo, la columna `id_personaje` se define como la clave primaria de la tabla `Personajes`, lo que significa que cada valor en esta columna será único y servirá para identificar de manera única a cada personaje en la tabla.

4.2. DEFAULT

La restricción DEFAULT se utiliza para establecer un valor predeterminado para una columna cuando no se proporciona un valor en una instrucción INSERT. Esto asegura que la columna tenga siempre un valor, incluso si no se especifica explícitamente durante la inserción de datos.

Ejemplo:

```
CREATE TABLE Tecnicas (
    id_tecnica INT PRIMARY KEY,
    nombre VARCHAR(100),
    descripcion VARCHAR(255),
    tipo ENUM('Ataque', 'Defensa', 'Soporte') DEFAULT 'Ataque',
    nivel_poder INT DEFAULT 0
);
```

En este ejemplo, la columna `tipo` de la tabla `Tecnicas` se establece con un valor predeterminado de `'Ataque'`, lo que significa que si no se proporciona un valor para esta columna durante la inserción de datos, se utilizará automáticamente `'Ataque'`. De manera similar, la columna `nivel_poder` se establece con un valor predeterminado de `0` si no se proporciona un valor durante la inserción.

Consideraciones Adicionales:

- La restricción PRIMARY KEY es fundamental para garantizar la integridad de los datos y proporcionar un identificador único para cada fila en una tabla.
- La restricción DEFAULT es útil para proporcionar valores predeterminados para las columnas, lo que facilita la inserción de datos cuando no se proporcionan valores específicos.
- Es importante elegir cuidadosamente las columnas que se definirán como PRIMARY KEY y determinar valores predeterminados adecuados para las columnas que los requieran.

12.1. Ejercicio de Restricciones.

En este ejercicio, se deberá crear una base de datos para almacenar información sobre los torneos de Dragon Ball. Se deberán aplicar diferentes tipos de restricciones, como CONSTRAINT, UNIQUE, CHECK, REFERENCES, ON DELETE, ON UPDATE, PRIMARY KEY y DEFAULT, para garantizar la integridad de los datos y la coherencia de la base de datos.

Pasos:

1. **Creación de la base de datos:**
   - Crear una nueva base de datos llamada `TorneosDragonBall`.
2. **Creación de la tabla Torneos:**
   - Crear una tabla llamada `Torneos` con las siguientes columnas:
     - `id_torneo`: (clave principal, entero y autoincremental).
     - `nombre`: (cadena de 100, única).
     - `ubicacion`: (cadena de 100).
     - `fecha_inicio`: (tipo fecha)
     - `fecha_fin`: (tipo fecha, por defecto null).
3. **Creación de la tabla Participantes:**
   - Crear una tabla llamada `Participantes` con las siguientes columnas:
     - `id_participante`: (clave principal, entero y autoincremental).
     - `nombre`: (cadena de 100).
     - `edad`: (entero y comprobar que es mayor que 18 y menor que 150)
     - `raza`: (cadena de 30).
     - `id_torneo`: (entero, fk de id_torneo, borrado y actualizado en cascada)
4. **Creación de datos de ejemplo:**
   - Insertar al menos tres torneos en la tabla `Torneos` y tres participantes en la tabla `Participantes`, asegurándose de que las restricciones UNIQUE y CHECK se respeten.
   - Asignar algunos participantes a los torneos existentes, asegurándose de que se respeten las restricciones FOREIGN KEY.
5. **Pruebas y consultas:**
   - Realizar consultas SELECT para verificar que los datos se hayan insertado correctamente en ambas tablas.
   - Realizar una actualización de la ubicación de un torneo en la tabla `Torneos` y verificar que los participantes asociados se actualicen correctamente debido a la restricción ON UPDATE CASCADE.
   - Realizar una eliminación de un torneo en la tabla `Torneos` y verificar que los participantes asociados se eliminen automáticamente debido a la restricción ON DELETE CASCADE.

12.2. EJERCICIO FINAL. Registro de Caballeros del Zodiaco con Constelaciones

Ejercicio: Registro de Caballeros del Zodiaco con Constelaciones

Objetivo:

Crear una base de datos simple para almacenar información sobre los Caballeros del Zodiaco, incluyendo sus nombres, signos zodiacales, constelaciones asociadas y detalles de las constelaciones.

Pasos a seguir:

1. **Creación de la Base de Datos:**
   - Utiliza la instrucción `CREATE DATABASE` para crear una nueva base de datos llamada "CaballerosDelZodiaco".
2. **Creación de la Tabla de Caballeros:**
   - Crea una tabla llamada "Caballeros" con los siguientes campos:
     - ID: Identificador único para cada Caballero (clave primaria).
     - Nombre: Nombre del Caballero.
     - SignoZodiacal: Signo zodiacal del Caballero (por ejemplo, Aries, Tauro, Géminis, etc.).
     - ConstelacionID: Clave foránea que hace referencia al ID de la constelación asociada (FK).
   - Utiliza la instrucción `CREATE TABLE` para definir la estructura de la tabla.
3. **Inserción de Registros de Caballeros:**
   - Inserta al menos cinco registros de Caballeros del Zodiaco en la tabla "Caballeros", incluyendo información sobre su nombre, signo zodiacal y el ID de la constelación asociada.
4. **Creación de la Tabla de Constelaciones:**
   - Crea una tabla llamada "Constelaciones" con los siguientes campos:
     - ID: Identificador único para cada Constelación (clave primaria).
     - Nombre: Nombre de la constelación.
     - Descripcion: Descripción de la constelación.
   - Utiliza la instrucción `CREATE TABLE` para definir la estructura de la tabla.
5. **Inserción de Registros de Constelaciones:**
   - Inserta al menos cinco registros de constelaciones en la tabla "Constelaciones", incluyendo información sobre el nombre y una breve descripción de cada constelación.
6. **Establecimiento de la Relación:**
   - Asegúrate de que el campo ConstelacionID en la tabla "Caballeros" sea una clave foránea que haga referencia al ID de la constelación en la tabla "Constelaciones".
7. **Consulta de Datos:**
   - Realiza consultas SELECT para verificar que los datos se han insertado correctamente en ambas tablas.
   - Por ejemplo, puedes consultar los Caballeros y las constelaciones asociadas, las características de una constelación específica, etc.

Registros para la Tabla de Caballeros:

1. Caballero: Pegaso
   - ID: 1
   - Nombre: Seiya
   - Signo Zodiacal: Sagitario
   - ConstelacionID: 3
2. Caballero: Dragón
   - ID: 2
   - Nombre: Shiryu
   - Signo Zodiacal: Capricornio
   - ConstelacionID: 4
3. Caballero: Cisne
   - ID: 3
   - Nombre: Hyoga
   - Signo Zodiacal: Acuario
   - ConstelacionID: 5
4. Caballero: Fénix

- ID: 4
- Nombre: Ikki
- Signo Zodiacal: Escorpio
- ConstelacionID: 6

5. Caballero: Andrómeda

   - ID: 5
   - Nombre: Shun
   - Signo Zodiacal: Piscis
   - ConstelacionID: 7

Registros para la Tabla de Constelaciones:

1. Constelación: Pegaso

   - ID: 1
   - Nombre: Pegaso
   - Descripción: Una constelación celestial en forma de caballo alado.

2. Constelación: Dragón

   - ID: 2
   - Nombre: Dragón
   - Descripción: Una de las constelaciones más grandes y prominentes del cielo nocturno.

3. Constelación: Cisne

   - ID: 3
   - Nombre: Cisne
   - Descripción: Representa a un cisne que vuela a lo largo de la Vía Láctea.

4. Constelación: Fénix

   - ID: 4
   - Nombre: Fénix
   - Descripción: Un ave mítica que renace de sus propias cenizas.

5. Constelación: Andrómeda

   - ID: 5
   - Nombre: Andrómeda
   - Descripción: Una constelación que representa a la princesa de la mitología griega, encadenada a una roca como sacrificio.

Modificaciones a realizar.

1. **Modificar el Campo 'SignoZodiacal' por 'Zodiaco' en la Tabla de Caballeros:**

   - Cambia el campo 'Signo Zodiacal' en la tabla de Caballeros para reflejar la última actualización de los signos zodiacales según la astrología moderna.

2. **Agregar un Nuevo Campo en la Tabla de Constelaciones:**

   - Agrega un nuevo campo llamado 'Estrella Principal' en la tabla de Constelaciones para indicar la estrella más brillante dentro de cada constelación.

3. **Cambiar el Nombre de la Tabla de Caballeros:**

   - Modifica el nombre de la tabla de Caballeros de 'Caballeros' a 'Caballeros_del_Zodiaco' para reflejar mejor el contexto del conjunto de datos.

4. **Añadir una Restricción de NOT NULL al Campo 'Nombre' en la Tabla de Caballeros:**

   - Asegúrate de que el campo 'Nombre' en la tabla de Caballeros no pueda contener valores nulos para garantizar la integridad de los datos.

5. **Modificar la Descripción en la Tabla de Constelaciones:**

   - Actualiza la descripción de la constelación 'Andrómeda' en la tabla de Constelaciones para incluir más detalles sobre su mitología y posición en el cielo:
   - "*Andrómeda es una constelación del hemisferio norte, cerca del Polo Norte Celeste, conocida por su vínculo con la mitología griega y la galaxia de Andrómeda (M31), una de las más grandes y brillantes observables desde la Tierra*."

Solución al Ejercicio completo.

12.3. EJERCICIO ERRORES.

Dado el siguiente archivo sql con errores, cárgalo en tu MySql, e identifica cuántos y cuáles errores hay y sus posibles soluciones.

Deberás subir un zip con el archivo .sql que lleva a cabo la creación de la base de datos solicitada con el nombre **Ejercicio1_DDL_Nombre_Apellido.sql** y el archivo de explicación correspondiente de nombre **con el mismo nombre pero en formato pdf.**

Comprueba su validez antes de subirlo. Si devuelve cualquier tipo de error al ejecutarse será penalizado como un **NO_APTO.**

12.4. EJERCICIO PUBS.

Se dispone de la siguiente Base de Datos para gestionar la información de los pubs de una determinada provincia.

PUB
#COD_PUB
NOMBRE
LICENCIA_FISCAL
DOMICILIO
FECHA_APERTURA
HORARIO
COD_LOCALIDAD

TITULAR
#DNI_TITULAR
NOMBRE
DOMICILIO
COD_PUB

EMPLEADO
#DNI_EMPLEADO
NOMBRE
DOMICILIO

EXISTENCIAS
#COD_ARTICULO
NOMBRE
CANTIDAD
PRECIO
COD_PUB

LOCALIDAD
#COD_LOCALIDAD
NOMBRE

PUB_EMPLEADO
#COD_PUB
#DNI_EMPLEADO
#FUNCION

Se pide escribir los comandos SQL que permitan la creación de las tablas anteriores teniendo en cuenta las siguientes restricciones:

1. Todos los valores son de tipo carácter excepto los campos FECHA_APERTURA (fecha), CANTIDAD, PRECIO y COD_LOCALIDAD (numéricos).
2. Los únicos campos que no son obligatorios son los campos DOMICILIO.
3. Los valores del campo horario sólo pueden ser HOR1, HOR2 y HOR3.
4. No es posible dar de alta EXISTENCIAS a precio 0.
5. El campo función de la tabla PUB_EMPLEADO sólo puede tener los valores CAMARERO, SEGURIDAD, LIMPIEZA.
6. Se ha de mantener la integridad referencial entre las tablas.
7. Las claves primarias vienen marcadas con el símbolo #.

Deberás subir el archivo .sql que lleva a cabo la creación de la base de datos solicitada con el nombre **Ejercicio2_DDL_Nombre_Apellido.sql.**

Comprueba su validez antes de subirlo. Si devuelve cualquier tipo de error al ejecutarse será penalizado como un **NO_APTO.**

12.5. EJERCICIO DE TALLERES

1. Crea las siguientes tablas:

Tabla COCHES:

Esta tabla almacenará información sobre los coches que llegan al taller.

- `mat`: Matrícula del coche, una cadena de 8 caracteres, será la clave primaria de esta tabla.
- `marca`: Marca del coche, una cadena de hasta 15 caracteres.
- `an_fab`: Año de fabricación del coche, un número de 4 dígitos.
- `modelo`: Modelo del coche, una cadena de hasta 15 caracteres.

Tabla MECANICOS:

Esta tabla almacenará información sobre los mecánicos que trabajan en el taller.

- `dni`: Número de identificación del mecánico, una cadena de 9 caracteres, será la clave primaria de esta tabla.
- `nombre`: Nombre del mecánico, una cadena de hasta 15 caracteres.
- `puesto`: Puesto del mecánico en el taller, una cadena de hasta 15 caracteres.
- `parcial`: Indicador de si el mecánico trabaja a tiempo parcial, un solo carácter.

Tabla TRABAJOS:

Esta tabla registrará los trabajos realizados en los coches.

- `mat`: Matrícula del coche, será una clave externa que hace referencia a la tabla COCHES.
- `dni`: Número de identificación del mecánico, será una clave externa que hace referencia a la tabla MECANICOS.
- `horas`: Número de horas empleadas en el trabajo, debe ser mayor de 0.5.
- `fecha_rep`: Fecha en que se realizó el trabajo.

Tabla CLIENTES:

Esta tabla almacenará información sobre los clientes del taller.

- `id_cliente`: Identificador del cliente, una cadena de 10 caracteres, será la clave primaria de esta tabla.
- `nombre`: Nombre del cliente, una cadena de hasta 50 caracteres.
- `telefono`: Número de teléfono del cliente, una cadena de hasta 15 caracteres.

Tabla PIEZAS:

Esta tabla almacenará información sobre las piezas utilizadas en los trabajos del taller.

- `id_pieza`: Identificador de la pieza, una cadena de 10 caracteres, será la clave primaria de esta tabla.
- `nombre`: Nombre de la pieza, una cadena de hasta 50 caracteres.
- `precio`: Precio de la pieza, un valor decimal.

Tabla FACTURAS:

Esta tabla registrará información sobre las facturas emitidas a los clientes del taller.

- `id_factura`: Identificador de la factura, una cadena de 10 caracteres, será la clave primaria de esta tabla.
- `id_cliente`: Identificador del cliente, una clave externa que hace referencia a la tabla CLIENTES.
- `fecha_emision`: Fecha de emisión de la factura.
- `total`: Total de la factura, un valor decimal.

2. Realiza las siguientes modificaciones sobre las tablas...

1. Una vez que las tablas básicas estén creadas, necesitaremos realizar algunas modificaciones adicionales para mejorar la estructura de la base de datos.

   ### Añadir atributo `modelo` a la tabla COCHES:

   Agregaremos un atributo `modelo` a la tabla COCHES para registrar el modelo de cada coche.

   ### Establecer mat y dni como llaves primarias de TRABAJOS:

Haremos que las columnas `mat` y `dni` en la tabla TRABAJOS formen una clave primaria compuesta, lo que asegurará que cada combinación de matrícula y número de identificación del mecánico sea única en la tabla.

Establecer dni de TRABAJOS como llave ajena respecto a MECANICOS:

Estableceremos la columna `dni` en la tabla TRABAJOS como una clave ajena que hace referencia a la columna `dni` en la tabla MECANICOS. Esto garantizará que solo se puedan asignar trabajos a mecánicos que estén registrados en la base de datos.

Reducir a 2 la longitud del atributo `an_fab` de la tabla COCHES:

Reducir la longitud del atributo `an_fab` en la tabla COCHES de 4 a 2 caracteres para permitir el almacenamiento de años con mayor precisión.

12.6. EJERCICIO BIBLIOTECA

Dadas las siguientes tablas:

Tabla Libro:

- `código`: Código del libro, una cadena de caracteres.
- `autor`: Autor del libro, una cadena de caracteres.
- `titulo`: Título del libro, una cadena de caracteres.
- `editor`: Editor del libro, una cadena de caracteres.
- `clase`: Clase del libro, una clave externa que hace referencia a la tabla Clase.
- `prestado`: Estado de préstamo del libro, un valor booleano (1 o 0).

Tabla Usuario:

- `dni`: Secuencia del usuario, una cadena de caracteres (por la letra).
- `nombre`: Nombre del usuario, una cadena de caracteres.
- `dirección`: Dirección del usuario, una cadena de caracteres.
- `fecha_ingreso`: Fecha de ingreso del usuario, un valor de fecha.
- `sexo`: Sexo del usuario, un carácter.

Tabla Clase:

- `clave`: Clave de la clase, una cadena de caracteres.
- `tiempo_de_prestamo`: Tiempo de préstamo de la clase, un valor numérico.

Tabla Préstamo:

- `código`: Código del préstamo, una cadena de caracteres.
- `secuencia`: Secuencia del préstamo, una cadena de caracteres.
- `fecha_inicio`: Fecha de inicio del préstamo, un valor de fecha.

Haz las siguientes modificaciones sobre las tablas:

Una vez creadas las tablas, realizaremos las modificaciones especificadas.

Modificaciones en la tabla Usuario:

- **Añadir campo `sexo`**: Agregaremos un campo `sexo` a la tabla Usuario para registrar el sexo de cada usuario.

Modificaciones en la tabla Libro:

- **Añadir campo `índice`**: Agregaremos un campo `índice` a la tabla Libro de tipo entero para proporcionar un índice numérico único para cada libro.

Modificaciones en la tabla Préstamo:

- **Definir valor por defecto para `prestado`**: Estableceremos el valor por defecto de la columna `prestado` en 1, indicando que el libro está prestado por defecto.

Restricciones:

Establecer restricción NOT NULL para `fecha_inicio`: Haremos que la columna `fecha_inicio` no pueda ser nula, ya que es un campo obligatorio.