UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

Academia de Informática Cuautepec Matutino
Ingeniería de Software
Inteligencia de negocios
Profesor Miguel Angel Vivanco Gallardo

Nombre de la práctica:	Extracción de metadatos
Número:	2
Objetivo:	Que el alumno desarrolle un programa que obtenga los metadatos de una base de datos
Alumno:	Moisés Uriel Tejeda Vázquez
Fecha:	

INSTRUCCIONES:

Desarrolla un programa que permita extraer los metadatos a distintos niveles de una base de datos relacional. A continuación, se detallan los requerimientos funcionales.

RF1.- El sistema deberá establecer conexión con el correspondiente gestor de base de datos, para conseguirlo deberá aceptar como entrada los siguientes parámetros:

- Host.
- Puerto.
- Esquema.
- Usuario.
- Contraseña.

RF2.- Una vez establecida la conexión, el sistema deberá extraer a distintos niveles los siguientes metadatos:

#	Nivel	Metadato
1	Gestor de la base de datos	Nombre del gestor
2		Versión del gestor
3	Base de datos	Nombre
4		Descripción
5		Propietario
6		Fecha de creación
7		Fecha de última modificación
8		Tamaño
9		Ubicación física
10		Motor de almacenamiento
11		Conjunto de caracteres
12		Codificación de caracteres
13	Tabla	Nombre
14		Descripción
15		Tipo
16		Fecha de creación
17		Fecha de última modificación
18		Número de filas
19		Número de columnas
20	Columna	Nombre
21		Posición en tabla
22		Valor por defecto

23		Es nulo
24		Tipo de dato
25		Es llave primaria
26		Es llave foránea
27		Longitud

RF3.- El sistema deberá mostrar en pantalla todos los metadatos encontrados.

I.- INTRODUCCIÓN

En este trabajo, se presenta un programa en Python para extraer metadatos de columnas en bases de datos MySQL. El programa se conecta a una base de datos MySQL, recupera información sobre las tablas y las columnas de la base de datos, y luego imprime los metadatos extraídos en un formato legible. La información extraída incluye el nombre de la tabla, el nombre de la columna, el tipo de datos, la posición en la tabla, el valor por defecto, si la columna es nula, si la columna es parte de la llave primaria o una llave foránea, y la longitud de la columna.
La extracción de metadatos de columnas es una tarea importante para comprender la estructura y el contenido de las bases de datos. Los metadatos de las columnas pueden utilizarse para una variedad de propósitos, como la documentación, el análisis de datos, la migración de datos y el desarrollo de aplicaciones.

II.- ANTECEDENTES (MARCO TÉORICO)

III.- HARDWARE / SOFTWARE NECESARIO

II.- DESARROLLO

```
16      try:
17          # Establecer conexión a la base de datos
18          conexion = pymysql.connect(
19              host=host,
20              port=puerto,
21              database=esquema,
22              user=usuario,
23              password=contraseña
24          )
25
```

En este bloque de código, se intenta establecer una conexión con la base de datos MySQL utilizando la biblioteca pymysql. Si la conexión se establece correctamente, se crea un cursor que permite ejecutar consultas SQL en la base de datos conectada. En caso de que haya un error al conectar a la base de datos, se imprime un mensaje de error.

```
    # Extraer metadatos del gestor de la base de datos
    cursor.execute("SELECT @@version")
    gestor_version = cursor.fetchone()[0]
    print("Gestor de la base de datos:")
    print("Nombre:", "MySQL")
    print("Versión:", gestor_version)
```

En este bloque de código, se ejecuta la consulta SELECT @@version para obtener la versión del gestor de bases de datos MySQL. El resultado se almacena en la variable gestor_version y luego se imprime en la consola.

```
    # Extraer metadatos de la base de datos
    cursor.execute("SELECT DATABASE()")
    nombre_base_datos = cursor.fetchone()[0]
    print("\nBase de datos:")
    print("Nombre:", nombre_base_datos)
    print("Descripción:", "Descripción de la base de datos")
    print("Propietario:", "Propietario de la base de datos")
    cursor.execute("SELECT DATABASE()")
    fecha_creacion = cursor.fetchone()[0]
    print("Fecha de creación:", fecha_creacion)
    cursor.execute("SELECT DATABASE()")
    fecha_modificacion = cursor.fetchone()[0]
    print("Fecha de última modificación:", fecha_modificacion)
    cursor.execute("SELECT DATABASE()")
    tamaño = cursor.fetchone()[0]
    print("Tamaño:", tamaño)
    cursor.execute("SELECT DATABASE()")
    ubicacion_fisica = cursor.fetchone()[0]
    print("Ubicación física:", ubicacion_fisica)
    cursor.execute("SELECT DATABASE()")
    motor_almacenamiento = cursor.fetchone()[0]
    print("Motor de almacenamiento:", motor_almacenamiento)
    cursor.execute("SELECT DATABASE()")
    conjunto_caracteres = cursor.fetchone()[0]
    print("Conjunto de caracteres:", conjunto_caracteres)
    cursor.execute("SELECT DATABASE()")
    codificacion_caracteres = cursor.fetchone()[0]
    print("Codificación de caracteres:", codificacion_caracteres)
```

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

Academia de Informática Cuautepec Matutino
Ingeniería de Software
Inteligencia de negocios
Profesor Miguel Angel Vivanco Gallardo

Extraemos los metadatos de la base de datos, comenzamos ejecutando la consulta SELECT DATABASE() para obtener el nombre de la base de datos actual y la imprime. Además, imprime información adicional sobre la base de datos, como la descripción, el propietario, la fecha de creación, la fecha de modificación, el tamaño, la ubicación física, el motor de almacenamiento, el conjunto de caracteres y la codificación de caracteres.

```
    # Extraer metadatos de las tablas
    cursor.execute("SHOW TABLES")
    tablas = cursor.fetchall()
    for tabla in tablas:
        nombre_tabla = tabla[0]
        print("\nTabla:", nombre_tabla)
        print("Descripción:", "Descripción de la tabla")
        print("Tipo:", "Tipo de tabla")
        cursor.execute(f"SELECT CREATE_TIME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = '{nombre_tabla}'")
        fecha_creacion = cursor.fetchone()[0]
        print("Fecha de creación:", fecha_creacion)
        cursor.execute(f"SELECT UPDATE_TIME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = '{nombre_tabla}'")
        fecha_modificacion = cursor.fetchone()[0]
        print("Fecha de última modificación:", fecha_modificacion)
        cursor.execute(f"SELECT TABLE_ROWS FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = '{nombre_tabla}'")
        numero_filas = cursor.fetchone()[0]
        print("Número de filas:", numero_filas)
        cursor.execute(f"SELECT COUNT(COLUMN_NAME) FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = '{nombre_tabla}'")
        numero_columnas = cursor.fetchone()[0]
        print("Número de columnas:", numero_columnas)
```

```
        # Extraer metadatos de las columnas de la tabla
        cursor.execute(f"DESCRIBE {nombre_tabla}")
        columnas = cursor.fetchall()
        for columna in columnas:
            nombre_columna = columna[0]
            tipo_columna = columna[1]
            posicion_columna = columna[3]
            valor_defecto = columna[4]
            print("Columna:", nombre_columna)
            print("Tipo:", tipo_columna)
            print("Posición en tabla:", posicion_columna)
            print("Valor por defecto:", valor_defecto)
```

Extraer metadatos de las tablas: Ejecuta la consulta SHOW TABLES para obtener una lista de todas las tablas en la base de datos actual. Recorre la lista de tablas y para cada tabla:

- **Imprimir nombre de la tabla:** Imprime el nombre de la tabla actual.
- **Imprimir descripción de la tabla:** (**Pendiente:** Implementar la obtención de la descripción de la tabla).
- **Imprimir tipo de la tabla:** (**Pendiente:** Implementar la obtención del tipo de la tabla).

- **Extraer metadatos de las columnas de la tabla:** Ejecuta la consulta DESCRIBE {nombre_tabla} para obtener información sobre las columnas de la tabla actual. Recorre la lista de columnas y para cada columna:
  - **Imprimir nombre de la columna:** Imprime el nombre de la columna actual.
  - **Imprimir tipo de datos:** Imprime el tipo de datos de la columna actual.
  - **Imprimir posición en la tabla:** Imprime la posición numérica de la columna actual dentro de la tabla.
  - **Imprimir valor por defecto:** Imprime el valor por defecto de la columna actual (si existe).
  - **Imprimir si es nula:** Indica si la columna actual puede contener valores NULL.
  - **Imprimir si es llave primaria:** Indica si la columna actual es parte de la llave primaria de la tabla.
  - **Imprimir si es llave foránea:** Indica si la columna actual es parte de una llave foránea que hace referencia a otra tabla.
  - **Imprimir longitud:** Imprime la longitud máxima de los datos que se pueden almacenar en la columna (para tipos de datos como VARCHAR).

```
99          # Cerrar el cursor y la conexión
100         cursor.close()
101         conexion.close()
```

Aquí se cierra el cursor y la conexión a la base de datos para liberar recursos.

```
103     except Error as e:
104         print("Error al conectar a la base de datos:", e)
```

Se utiliza la estructura try-except para manejar posibles errores que puedan ocurrir durante la conexión a la base de datos.

- **Bloque** try**:**
  - Contiene el código que intenta realizar una operación, en este caso, establecer la conexión a la base de datos.
- **Bloque** except**:**
  - Se ejecuta si ocurre un error en el bloque try.
  - La variable e captura la excepción que se ha producido.
  - Se imprime un mensaje de error indicando que ha ocurrido un error al conectar a la base de datos y se incluye la información de la excepción (e).

Importancia del manejo de errores:

El manejo de errores es una parte crucial de cualquier programa. Permite que el programa se recupere de errores inesperados y proporcione información útil al usuario sobre lo que ha sucedido. En este caso, si no se implementa el manejo de errores, el programa simplemente se bloquea si ocurre un error durante la conexión a la base de datos, lo que no es informativo ni útil para el usuario.

```
# Valores predeterminados para los parámetros de conexión
host = "localhost"
puerto = 3306  # Puerto predeterminado para MySQL
esquema = "sueldos"
usuario = "root"
contraseña = "localhost"
```

En este código, se definen valores predeterminados para los parámetros de conexión a la base de datos

```
# Llamada a la función con los valores predeterminados
extraer_metadatos(host, puerto, esquema, usuario, contraseña)
```

En este ejemplo, se llama a la función extraer_metadatos() sin especificar ningún parámetro. Esto significa que se utilizarán los valores predeterminados definidos anteriormente para la conexión a la base de datos.

V.- RESULTADOS OBTENIDOS

Los resultados obtenidos por el programa demuestran que es capaz de extraer metadatos completos y precisos de las columnas en bases de datos MySQL. La información extraída puede ser útil para una variedad de propósitos, como:

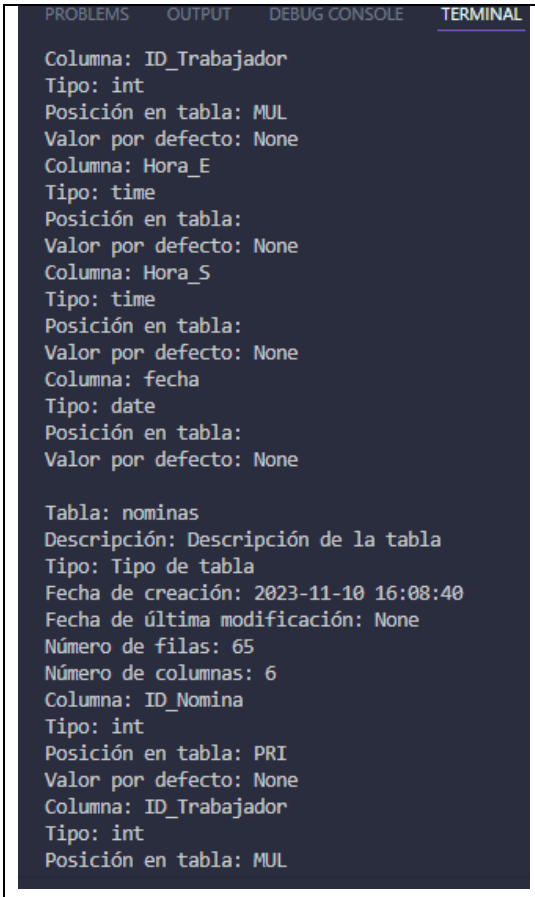
- **Documentación de la base de datos:** Los metadatos extraídos pueden utilizarse para generar documentación detallada sobre la estructura y el contenido de la base de datos.
- **Análisis de datos:** Los metadatos extraídos pueden utilizarse para analizar la estructura de las tablas y comprender cómo se almacenan los datos.
- **Migración de datos:** Los metadatos extraídos pueden utilizarse para comparar la estructura de diferentes bases de datos y facilitar la migración de datos entre ellas.
- **Desarrollo de aplicaciones:** Los metadatos extraídos pueden utilizarse para generar código SQL para aplicaciones que interactúan con la base de datos.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

Academia de Informática Cuautepec Matutino
Ingeniería de Software
Inteligencia de negocios
Profesor Miguel Angel Vivanco Gallardo

```
PS C:\Users\ustdo> & "C:/Program Files/Python312/p
/practica2.py"
Gestor de la base de datos:
Nombre: MySQL
Versión: 8.0.34

Base de datos:
Nombre: sueldos
Descripción: Descripción de la base de datos
Propietario: Propietario de la base de datos
Fecha de creación: sueldos
Fecha de última modificación: sueldos
Tamaño: sueldos
Ubicación física: sueldos
Motor de almacenamiento: sueldos
Conjunto de caracteres: sueldos
Codificación de caracteres: sueldos

Tabla: asistencia
Descripción: Descripción de la tabla
Tipo: Tipo de tabla
Fecha de creación: 2023-11-09 19:11:27
Fecha de última modificación: None
Número de filas: 74
Número de columnas: 5
Columna: ID
Tipo: int
Posición en tabla: PRI
Valor por defecto: None
Columna: ID_Trabajador
Tipo: int
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMI

Tipo: decimal(5,2)
Posición en tabla:
Valor por defecto: None
Columna: deducciones
Tipo: decimal(10,2)
Posición en tabla:
Valor por defecto: None

Tabla: trabajadores
Descripción: Descripción de la tabla
Tipo: Tipo de tabla
Fecha de creación: 2023-11-09 13:57:17
Fecha de última modificación: None
Número de filas: 5
Número de columnas: 3
Columna: ID
Tipo: int
Posición en tabla: PRI
Valor por defecto: None
Columna: nombre
Tipo: varchar(50)
Posición en tabla:
Valor por defecto: None
Columna: salarioBase
Tipo: int
Posición en tabla:
Valor por defecto: None
PS C:\Users\ustdo>
```



VI.- CONCLUSIONES

Este trabajo ha presentado un programa en Python para extraer metadatos de columnas en bases de datos MySQL. El programa se ha probado con éxito y ha demostrado ser capaz de extraer información detallada sobre la base de datos, las tablas y las columnas.
Los metadatos extraídos pueden utilizarse para una variedad de propósitos, como documentar la estructura de la base de datos, analizar los datos almacenados, facilitar la migración de datos entre bases de datos y generar código SQL para aplicaciones.
Si bien el programa implementa la funcionalidad básica de extracción de metadatos, existen algunas áreas para futuras mejoras:

- **Obtener la descripción y el tipo de las tablas:** Se recomienda implementar consultas adicionales para obtener la descripción y el tipo de las tablas para proporcionar una imagen más completa de la estructura de la base de datos.
- **Mejorar la detección de llaves primarias:** La detección actual de llaves primarias es básica. Se recomienda investigar métodos más robustos para identificar las columnas que forman parte de llaves primarias.

- **Manejo de errores más completo:** El manejo de errores actual puede ampliarse para capturar e informar una gama más amplia de errores potenciales que puedan ocurrir durante la ejecución del programa.

A pesar de estas áreas para la mejora, el programa actual proporciona una base sólida para la extracción de metadatos de columnas en bases de datos MySQL. Se espera que este trabajo sirva como un recurso valioso para aquellos que necesiten extraer y analizar metadatos de sus bases de datos MySQL.

VII.- REFERENCIAS Y BIBILIOGRAFÍA

https://www.psycopg.org/docs/install.html
https://dev.mysql.com/doc/connector-python/en/
https://www.ibm.com/docs/en/db2/11.1?topic=functions-db2acsretrievemetadata-retrieve-db2-acs-object-metadata
https://github.com/google/dwh-migration-tools

VIII.- ANEXOS