Proyecto Final de la materia Bases de Datos I

Implementación mínima del proyecto

Entrega

La entrega consiste subir a una carpeta del GIT del grupo de al menos cuatro archivos:

- **Justificación del diseño.** Este archivo puede ser un documento estándar (Mardown o PDF) donde se justifique el diseño de la base de datos que debe corresponder al menos hasta 3NF, de forma similar al TP5.
- **Fuente SQL.** Este archivo debe contener **todo** el código SQL necesario para crear las tablas y cargar datos inicales para el testing del proyecto (*e.g.* INSERT's)
- **Archivo de código Python.** Este archivo debe corresponder con las consignas que se detallan abajo el proyecto elegido por el grupo.
- **Archivo de Diseño de Esquema de Tablas.** Puede ser una captura de la imagen del diseño (*e.g.* imagen JPG, PNG, etc) o alternativamente puede ser un archivo DBML con el diseño.

Características Mínimas Exigidas

El proyecto debe contener **cada una** de estas caracteríticas mínimas.

- **Diseño esperado.** Identifica entidades fuertes y débiles, atributos, relaciones y cardinalidades.
- **Normalización:** Debes aplicar las formas normales (1NF, 2NF y 3NF) para optimizar el diseño de la base de datos y eliminar redundancias.
- **Restricciones de Integridad:** Define llaves primarias y foráneas. Aplica restricciones `NOT NULL`, `UNIQUE`.
- **Operaciones en cascada** Define operaciones en cascada convenientes usando `ON UPDATE CASCADE`, `ON DELETE RESTRICT`, etc. donde sea necesario para prevenir errores de consistencia.
- **Consultas Avanzadas:** Implementa `INNER JOIN`, `LEFT JOIN` y `RIGHT JOIN` para recuperar datos relacionados. Utiliza cunando sea necesario funciones agregadas y cláusulas como `GROUP BY`, `HAVING`, `ORDER BY`.
- **Transacciones y Manejo de Errores:** Utiliza transacciones para garantizar la consistencia de los datos en operaciones críticas.
- **Procedimientos Almacenados y Funciones:** Crea procedimientos para manejar operaciones complejas como préstamos y devoluciones. Implementa funciones para cálculos específicos, como multas por retraso.
- **Índices:** Crea índices para mejorar el rendimiento de las consultas más frecuentes. El diseño debe contener al menos *un* (1) índice.
- **Python y `mysql.connector`:** Desarrolla una aplicación en Python que interactúe con la base de datos MySQL. Implementa una interfaz de usuario (CLI) para facilitar la interacción con la aplicación.

Proyecto 3: Sistema de Gestión de Hospital

Descripción:

Desarrolla un sistema para gestionar un hospital que incluya pacientes, médicos y turnos de consulta médica.

- **Modelo del Sistema.** Representa entidades como Pacientes, Médicos y Turnos con fecha y horario.
- **Datos inciales** La base de datos inicial debe contener al menos diez (10) pacientes y diez (10) médicos. La cantidad *promedio* de los turnos debe ser de diez (10) turnos por cada paciente.

Ejercicio: Consignas para el Menú del CLI

1. **Gestión de Pacientes**: Registrar, actualizar, ver y eliminar información de pacientes.
2. **Gestión de Doctores**: Agregar, actualizar y ver detalles de los doctores, incluyendo especialidades.
3. **Manejo de Turnos**: Programar, actualizar o cancelar turnos.
4. **Búsquedas Avanzadas**: Recuperar pacientes o médicos mediante diferentes atributos (nombre, especialidad, ID de paciente). Tener en cuanta que la búsqueda puede ser por parte del texto a buscar.
5. **Reporte de turnos**: Mostrar un reporte donde muestre los tres (3) médicos que más turnos tienen y la cantidad de turnos de cada uno.
6. **Cancelación de turnos**: Se debe poder cancelar para un médico **dado** y un rango de fechas **dado**.