Estandarización del Diseño de Base de Datos para el Sistema de Citas Médicas

Para garantizar la calidad, mantenimiento y escalabilidad de la base de datos del sistema de gestión de citas médicas, es esencial adoptar una serie de reglas y convenciones al definir la estructura y los nombres de los objetos dentro de la base de datos. A continuación, se detallan las principales directrices para el diseño y gestión de la base de datos.

1. Convenciones de Nombres

Formato de Nombres (Snake Case):
Se debe utilizar el estilo *snake_case* para todos los nombres de objetos en la base de datos. Esto implica que las palabras se separan por guiones bajos y todo el texto debe ir en minúsculas.

Ejemplo:

- nombre_paciente
- fecha_cita

Nombres de Tablas:
Las tablas deben tener nombres **en plural** para reflejar que representan colecciones de entidades. Se debe utilizar la primera letra en mayúscula para mejorar la legibilidad y claridad.

Ejemplo:

- Pacientes
- Doctores
- Citas
- Especialidades

Nombres de Columnas:
Las columnas deben tener nombres **descriptivos** y estar en **minúsculas**, utilizando *snake_case* para mejorar la consistencia.

Ejemplo:

- paciente_id
- doctor_id
- fecha_cita
- especialidad_id

Índices:
Los índices deben llevar el prefijo idx_ seguido del nombre de la tabla y las columnas involucradas.

Ejemplo:

- idx_citas_fecha

- idx_pacientes_nombre

Vistas:
Las vistas deben comenzar con el prefijo vw_, seguidas de un nombre descriptivo que explique la función de la vista.

Ejemplo:

- vw_pacientes_activos
- vw_citas_por_doctor

Procedimientos Almacenados:
Los procedimientos almacenados deben llevar el prefijo sp_ y un nombre que describa su propósito de manera clara.

Ejemplo:

- sp_insertar_cita
- sp_actualizar_estado_cita

Funciones:
Las funciones deben comenzar con el prefijo fn_ y usar nombres que indiquen claramente su función.

Ejemplo:

- fn_obtener_historial_paciente
- fn_calcular_edad_paciente

2. Definición de Tipos de Datos y Estándares

Selección de Tipos de Datos:
Es necesario elegir los tipos de datos adecuados según la naturaleza de los datos almacenados:

- INT para identificadores numéricos, como paciente_id, doctor_id.
- VARCHAR para nombres y descripciones, asegurándose de no asignar tamaños excesivos.
- DATE para fechas, como fecha_cita.
- TIME para horas de cita, como hora_cita.
- BOOLEAN para campos que manejan valores binarios, como estado_cita.

Uso de NULL y NOT NULL:
Las columnas **NOT NULL** deben ser usadas en aquellas columnas que son esenciales para la consistencia de los datos. Las claves primarias deben ser **NOT NULL**, así como las columnas que no pueden tener valores vacíos, como las fechas y horas de citas.

Ancho de las Columnas:
Limitar el tamaño de las columnas VARCHAR según la necesidad real. Por ejemplo, para almacenar nombres de pacientes o doctores, se puede usar un tamaño de VARCHAR(100).

3. Llaves y Relaciones

Llaves Primarias:
Cada tabla debe tener una **clave primaria** única. Lo ideal es que sea de tipo INT o BIGINT y que sea **auto-incremental** para simplificar la inserción de datos.

Ejemplo:

- id_paciente
- id_doctor
- id_cita

Llaves Foráneas:
Las **llaves foráneas** deben ser utilizadas para mantener la integridad referencial entre tablas. Las relaciones entre tablas deben ser claras, usando el nombre de la tabla referenciada seguido de _id para las claves foráneas.

Ejemplo:

- doctor_id en la tabla citas, donde hace referencia a id_doctor en la tabla doctores.
- paciente_id en la tabla citas, donde hace referencia a id_paciente en la tabla pacientes.

4. Consultas y Estilo SQL

Estilo de Consultas:
Hay que utilizar **mayúsculas** para las palabras reservadas de SQL, como SELECT, FROM, WHERE, y JOIN, para mejorar la legibilidad.

Ejemplo:

```
SELECT nombre_paciente, fecha_cita

FROM citas

WHERE estado_cita = 'pendiente';
```

Alias de Tablas:
Las tablas deben ser referenciadas usando **alias cortos**, preferentemente con una sola letra, para mejorar la legibilidad, especialmente en consultas con múltiples uniones.

Ejemplo:

```
SELECT p.nombre_paciente, c.fecha_cita

FROM pacientes p

JOIN citas c ON p.id_paciente = c.paciente_id;
```

Consultas Complejas:
Hay que evitar subconsultas innecesarias. En su lugar, se deben usar JOIN para combinar tablas y mejorar el rendimiento.

Ejemplo:

```
SELECT p.nombre_paciente, d.nombre_doctor

FROM citas c

JOIN pacientes p ON c.paciente_id = p.id_paciente

JOIN doctores d ON c.doctor_id = d.id_doctor

WHERE c.fecha_cita = '2024-12-10';
```

Indentación:
Es necesario **indentar** las consultas correctamente, especialmente cuando se usan subconsultas y condiciones complejas, para mejorar la legibilidad.

Ejemplo:

```
SELECT nombre_paciente, fecha_cita

FROM citas

WHERE fecha_cita > '2024-11-01'

  AND estado_cita = 'confirmada';
```

5. Optimización y Rendimiento

Índices:
Hay que crear **índices** en columnas que se utilicen frecuentemente en filtros (WHERE), uniones (JOIN), o en cláusulas de ordenación (ORDER BY).

- **No duplicar índices:** No se debe crear un índice redundante si ya existe uno en la misma columna.
- **Usar índices con moderación:** Demasiados índices pueden ralentizar las operaciones de inserción y actualización.

Normalización:
Aplicar principios de **normalización** para reducir la redundancia de datos y garantizar la integridad. Se debe llevar el esquema de la base de datos, al menos, hasta la **Tercera Forma Normal (3NF)**.

6. Seguridad y Roles

Definición de Roles:
Los roles deben ser definidos según las necesidades de los usuarios dentro del sistema de gestión de citas médicas. Algunos ejemplos de roles son:

- **admin:** Acceso completo a todas las funcionalidades del sistema, incluyendo la gestión de usuarios y la configuración.
- **doctor:** Acceso a los detalles de las citas, pacientes, y la capacidad de actualizar el estado de las citas.
- **receptionist:** Acceso a las citas y la capacidad de gestionarlas (agendar, modificar).
- **read_only:** Solo lectura, para usuarios que necesiten consultar información sin hacer modificaciones.

Protección de Datos Sensibles:
Hay que asegurar la **encriptación de datos sensibles**, como los historiales médicos y las contraseñas. Los datos personales de los pacientes deben estar protegidos y no deben ser expuestos en vistas o procedimientos almacenados.

7. Gestión de Migraciones

Uso de Herramientas de Migración:
Es recomendable usar herramientas como **Liquibase** o **Flyway** para gestionar los cambios en el esquema de la base de datos y garantizar que las migraciones se apliquen de manera controlada.

Documentación de Cambios:
Es fundamental mantener un registro detallado de cada cambio estructural realizado en la base de datos, incluyendo fechas y descripciones claras.