UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

ABD Project (Sistema de Asistencia)

Administración de Bases de Datos

Nombre del proyecto: ABD Project

Integrantes del equipo: Mata Guerra David - 325797
(Agrega aquí a los demás integrantes)

Docente: (Nombre del Docente)

Fecha de Entrega: 2 de diciembre del 2025

Documentación General del Sistema de Asistencia.

1 Arquitectura del sistema gestor de base de datos

1.1 Características del DBMS utilizado

- Versión de PostgreSQL utilizada: 16.10 (instalada sobre Ubuntu 24.04 dentro de un contenedor Docker).
- Características relevantes del proyecto:
  - Motor relacional con soporte transaccional completo (ACID).
  - Soporte de claves primarias, foráneas, restricciones y vistas, utilizado para el modelado de la base de datos de asistencia.
  - Soporte de funciones y procedimientos almacenados en SQL, aprovechados en el archivo `procedimientos_asistencia.sql`.
  - Mecanismo de triggers, utilizado para automatizar lógica de negocio definida en `triggers_asistencia.sql`.
  - Mecanismo de índices configurables, documentados en `indices_asistencia.sql`, para optimizar consultas frecuentes.

1.2 Características del DBMS utilizado

- Versión de PostgreSQL utilizada: 16.10, ejecutándose sobre Ubuntu 24.04 dentro de un contenedor Docker.
- Se trata de un sistema gestor de bases de datos relacional, que permite trabajar con tablas relacionadas, vistas, índices y procedimientos almacenados.
- En este proyecto se aprovechan estas características para modelar y gestionar la base de datos de asistencia (tablas, índices, procedimientos, triggers y vistas definidos en los scripts de la carpeta `db-arquitecture/`).

1.3 Estructura de memoria y procesos de la instancia

- Explicación general de los procesos principales de PostgreSQL:
  - Un proceso maestro `postgres` que coordina la instancia del servidor.
  - Procesos auxiliares como *checkpointer*, *writer*, *wal writer* y *autovacuum*, encargados de escritura en disco, manejo del WAL y mantenimiento automático de tablas.
- Mención de los archivos/directorios importantes:
  - Directorio de datos: `/var/lib/postgresql/16/main` dentro del contenedor (configuración por defecto del paquete de Ubuntu).
  - Archivo de configuración principal: `postgresql.conf`, donde se definen parámetros como puerto, memoria compartida, autovacuum, etc.
  - Archivo de control de accesos: `pg_hba.conf`, donde se especifican los métodos de autenticación y orígenes permitidos.

1.4 Estructura de memoria y procesos de la instancia

- PostgreSQL se ejecuta como un servicio dentro del contenedor y mantiene varios procesos en segundo plano que se encargan de atender conexiones y guardar la información en disco.
- Los datos de la base se almacenan en un directorio interno de PostgreSQL (por defecto en `/var/lib/postgresql/16/main` dentro del contenedor).
- La configuración principal del servidor se realiza a través de los archivos `postgresql.conf` y `pg_hba.conf`, donde se ajustan parámetros como el puerto y las reglas de acceso.

1.5 Instalación y configuración básica

- Resumen del entorno donde corre la BD:
  - Sistema operativo base: Ubuntu 24.04 ejecutándose dentro de un contenedor Docker.
  - El contenedor se construye a partir del `Dockerfile` ubicado en `ubuntu-container/` y se orquesta mediante `docker-compose.yml`.
  - El servicio de PostgreSQL se instala mediante los paquetes `postgresql`, `postgresql-contrib` y `postgresql-client`.
- Variables de ambiente relevantes:
  - `DEBIAN_FRONTEND=noninteractive` para evitar prompts interactivos durante la instalación.
  - `TZ=UTC` para configurar la zona horaria del sistema dentro del contenedor.
- Configuraciones mínimas realizadas para que el sistema funcione:
  - Se modifica la contraseña del usuario `postgres` mediante: `ALTER USER postgres WITH PASSWORD 'postgres';`
  - El servicio PostgreSQL se expone en el puerto `5432` del contenedor y se mapea al puerto `5432` del host mediante Docker Compose.
  - El contenedor se mantiene en ejecución iniciando el servicio con `service postgresql start` y luego un comando de espera (`sleep infinity`) para permitir el acceso continuo durante las pruebas.

1.6 Instalación y configuración básica

- La base de datos se ejecuta en un contenedor Docker basado en Ubuntu 24.04, definido en la carpeta `ubuntu-container/` mediante un `Dockerfile` y un archivo `docker-compose.yml`.
- Durante la construcción de la imagen se instalan los paquetes de PostgreSQL y se configura la contraseña del usuario administrador `postgres` con el valor `postgres`.
- El puerto `5432` del contenedor se publica en el mismo puerto del equipo host, lo que permite conectar desde herramientas externas usando la dirección `localhost:5432`.

1.7 Clientes e IDEs de acceso

- Herramientas utilizadas:
  - `psql`, el cliente de línea de comandos de PostgreSQL, instalado dentro del contenedor y utilizado para ejecutar los scripts SQL del proyecto (inicialización, índices, procedimientos, triggers, vistas, seguridad, etc.).
  - Opcionalmente, un IDE gráfico como *pgAdmin* o *DBeaver* desde el host, conectándose a `localhost:5432` con el usuario `postgres` y contraseña `postgres`.
- Captura de pantalla de la conexión a la BD de asistencia:
  - Imagen

2 Diseñado de la base de datos

2.1 Modelo Entidad-Relación

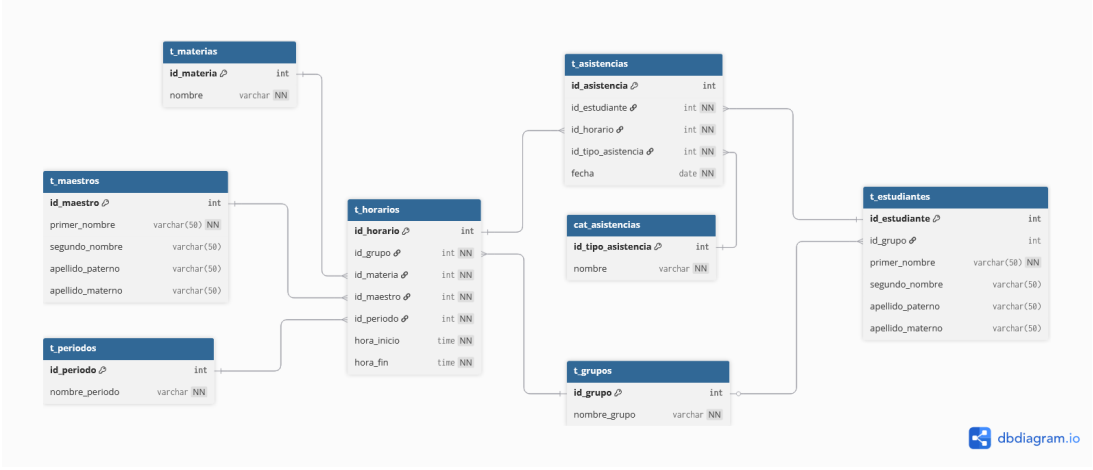

Figura 1: Diagrama Entidad-Relación de la base de datos de asistencia

2.2 Modelo Físico de la base de datos

Explicación lógica de las tablas

- t_estudiantes: almacena la información básica de los estudiantes, incluyendo su grupo asignado.
- t_maestros: almacena la información básica de los maestros.
- t_materias: contiene las materias disponibles en la institución.
- t_grupos: define los grupos o clases a los que pertenecen los estudiantes.
- t_periodos: registra los periodos académicos.
- t_horario: relaciona grupos, materias, maestros y periodos con horarios específicos.
- cat_asistencias: catálogo de tipos de asistencia (asistencia, retardo, falta, justificada, etc.).

- t_asistencias: registra las asistencias de los estudiantes por horario y fecha.

2.2.1. init_asistencia.sql

El script init_asistencia.sql contiene las instrucciones SQL para crear las tablas mencionadas anteriormente, junto con sus claves primarias, foráneas y restricciones necesarias para mantener la integridad referencial. Además, incluye las relaciones entre las tablas que reflejan el modelo entidad-relación diseñado para la base de datos de asistencia. Y finalmente población de la bd con un mínimo de

- 10 estudiantes
- 5 maestros
- 5 materias
- 3 grupos
- 1 periodo
- 1 semana de registros de asistencia

2.3 Optimización de la base de datos

Definicion de indices

- Índice para búsqueda rápida de asistencia por estudiante (igualdad).
  Se usa un índice **HASH** sobre t_asistencias(id_estudiante) para optimizar consultas de igualdad como: `SELECT * FROM t_asistencias WHERE id_estudiante = ?`. Los índices hash son ideales para lookups exactos.
- Índice para consultas por fecha (rangos temporales).
  Se usa un índice **BRIN** sobre t_asistencias(fecha) para acelerar rangos por día/semana/mes en tablas que crecen cronológicamente. BRIN es liviano y eficiente con datos correlacionados por bloque.
- Índice para consultas por grupo y materia (filtrado y joins vía horario).
  Se usa un índice **BTREE compuesto** sobre t_horarios(id_grupo, id_materia) para filtrar rápidamente combinaciones de grupo+materia y mejorar los joins posteriores con t_asistencias por id_horario.

indices_asistencia.sql

El script indices_asistencia.sql contiene las instrucciones SQL para crear los índices definidos anteriormente en la base de datos de asistencia. **Explicación de las decisiones tomadas en los índices:**

- **¿Por qué se eligió cada índice?**
  HASH para igualdad en id_estudiante; BRIN para rangos en fecha con crecimiento cronológico; BTREE compuesto para combinaciones frecuentes id_grupo+id_materia.
- **¿Qué tipo de consultas se benefician?**
  HASH: `WHERE id_estudiante = ?`. BRIN: `WHERE fecha BETWEEN ? AND ?` o por

día. BTREE compuesto: `WHERE id_grupo = ? AND id_materia = ?` y joins hacia t_asistencias por id_horario.

- **Impacto en inserciones/actualizaciones:**
  Los índices son primordiales en una buena base de datos, además el tipo de indice compuesto aunque no lo hayamos visto a profundidad en la clase, el maestro nos mencionó de su utilidad e importancia.

2.4 Estructuras avanzadas de la base de datos

Vistas

Procedimientos almacenados

Triggers

Índices

Se hace referencia a el apartado de indices en la seccion de optimizacion

3 Modelos de seguridad en bases de datos

3.1 Seguridad a nivel de usuarios y roles

Modelo de seguridad

- Roles mínimos:
  - rol_profesor: puede consultar sus grupos y registrar asistencias.
  - rol_control_escolar: puede consultar reportes globales, modificar estudiantes/grupos.
  - rol_consulta: solo lectura de vistas de reporte.
- Usuarios de ejemplo:
  - u_profesor_demo (asignado a rol_profesor).
  - u_control_escolar_demo (asignado a rol_control_escolar).
- Script seguridad_asistencia.sql
  - CREATE ROLE / CREATE USER o CREATE ROLE ... LOGIN.
  - GRANT y REVOKE a nivel base de datos, esquema, tablas y vistas. o Comentarios que expliquen qué puede y qué no puede hacer cada rol/usuario.

3.2 Backups y recuperación con cron

- Script de respaldo lógico automático en bash **Referenciar a el script backup_asistencia.sh**
- Cronjob para ejecutar el respaldo diario **Referenciar a el script cron_backup_asistencia.txt e implementar screenshot de la pagina para hacer crons**

- Script para restaurar desde un backup **Referenciar a el script restore_asistencia.sh**

4 Conclusiones

The...