UNIVERSIDAD DEL VALLE DE GUATEMALA

BASE DE DATOS 1

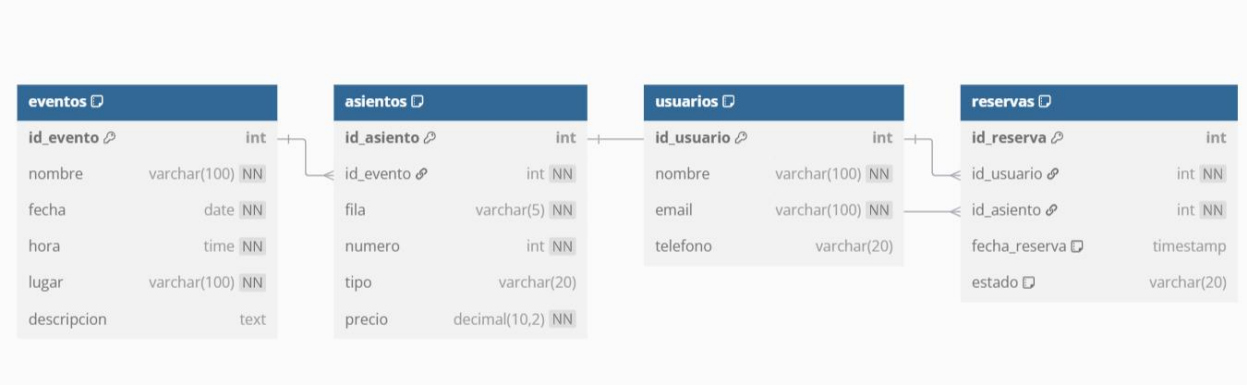
Sección 30

“Proyecto 2”

Eliazar José Pablo Canastuj Matías - 23384

GUATEMALA, 11 de abril de 2025

Fase 1: Diseño de la Base de Dato



Fase 2: Generación de Datos de Prueba

Archivo: data.sql

Fase 3: Implementación del Programa de Simulación

Link de GitHub: https://github.com/can23384/Proyecto2_BaseDeDatos1.git

📘 Manual de Uso – Simulador de Reservas Concurrentes

🧱 Requisitos Previos

1. PostgreSQL instalado
2. pgAdmin 4 funcionando correctamente
3. Python 3.8+ instalado
4. Librerías necesarias:

   psycopg2

   Instálalo con: bash pip install psycopg2

🔧 Paso 1: Crear la base de datos en pgAdmin 4

Abrir pgAdmin 4.

Click derecho en “Databases” > Create > Database.

Nombrar la base de datos: Proyecto_2.

Click en Save.

📩 Paso 2: Ejecutar ddl.sql y data.sql

En pgAdmin, hacer clic en la base Proyecto_2.

Ir a la pestaña Query Tool.

Cargar el archivo ddl.sql y ejecutarlo.

Luego, cargar y ejecuta data.sql.

🔄 Paso 3: Configurar el script de simulación en Python

Asegurarse de tener tu script proyecto2.py

Abrir el archivo y edita esta sección si es necesario:

```
CONFIG_BD = {
    'dbname': 'Proyecto_2',
    'user': 'postgres',       # Cambiar si el usuario es otro
    'password': '123456',     # Coloca la contraseña real
    'host': 'localhost',
    'port': 5432
}
```

▶️ Paso 4: Ejecutar la simulación

Desde la terminal, ejecutar el archivo Python:

python proyecto2.py

Se vera una interfaz interactiva:

🎛️ Simulador de Reservas Concurrentes

Selecciona el nivel de aislamiento:

1. READ COMMITTED

2. REPEATABLE READ

3. SERIALIZABLE

Opción [1-3]:

Si se quiere repetir una simulación, simplemente hay que volver a ejecutarlo.

Aumenta el número de usuarios o cambia el nivel de aislamiento para ver diferencias en el resultado.

Fase 4: Experimentación y Pruebas

Usuarios concurrentes	Nivel de aislamiento	Reservas exitosas	Reservas fallidas	Tiempo promedio
5	READ COMMITTED	5	0	153.77 ms
15	READ COMMITTED	11	4	127.89 ms
30	READ COMMITTED	15	15	138.13 ms
5	REPEATABLE READ	5	0	188.98 ms
15	REPEATABLE READ	11	5	141.24 ms
30	REPEATABLE READ	16	14	141.05 ms
5	SERIALIZABLE	5	0	161.84 ms
15	SERIALIZABLE	13	2	177.75 ms
30	SERIALIZABLE	20	10	180.19 ms

Fase 5: Análisis y Reflexión

Conclusiones sobre el manejo de concurrencia en bases de datos.

El manejo de concurrencia en bases de datos es fundamental para garantizar la integridad y consistencia de los datos cuando múltiples usuarios acceden o modifican la misma información simultáneamente. A través de la simulación de reservas concurrentes, se evidenció cómo las transacciones y los niveles de aislamiento controlan correctamente los conflictos de acceso, especialmente en escenarios donde varios usuarios compiten por los mismos recursos.

¿Cuál fue el mayor reto al implementar la concurrencia?

El principal reto fue asegurar que las transacciones fueran correctamente aisladas y sincronizadas usando SELECT ... FOR UPDATE, evitando así condiciones de carrera y duplicación de reservas.

¿Qué problemas de bloqueo encontraron?

Se observaron bloqueos cuando múltiples hilos intentaban acceder al mismo asiento al mismo tiempo, especialmente con niveles de aislamiento más altos. Sin embargo, PostgreSQL gestionó bien estos bloqueos, permitiendo que solo una transacción completara la reserva y las demás se rechazaran correctamente.

¿Cuál fue el nivel de aislamiento más eficiente?

READ COMMITTED ofreció mayor rapidez, pero con riesgo de lecturas no repetibles. SERIALIZABLE fue el más seguro y correcto, aunque más lento debido a su rigidez en la sincronización. REPEATABLE READ ofreció un buen equilibrio entre integridad y rendimiento.

¿Qué ventajas y desventajas tuvo el lenguaje seleccionado?

Ventajas: Python es simple, legible y tiene excelentes bibliotecas como psycopg2 para manejo de PostgreSQL. También facilita la creación de simulaciones concurrentes con threading.

Desventajas: El modelo de hilos de Python no aprovecha completamente múltiples núcleos, y no es ideal para tareas muy intensivas en CPU. Además, el manejo de excepciones en múltiples hilos requiere especial cuidado.