Universidad del Valle de Guatemala
Facultad de Ingeniería
Ciencia de la Computación y Tecnologías de la información
cc3088 - Bases de Datos 1 Ciclo 1, 2025

Laboratorio 3

Instrucciones Generales

- Este laboratorio debe realizarse en grupos de 2 personas.
- Deben usar un ORM (Object-Relational Mapping) de su elección (Eloquent, SQLAlchemy, Sequelize, Prisma, etc.).
- Deberán construir una aplicación CRUD completa para una entidad principal que tenga múltiples elementos relacionados a través de una tabla intermedia.
- La aplicación debe permitir crear, editar y eliminar registros, así como mostrar un índice completo a través de una vista (VIEW) en la base de datos.
- Entregar los siguientes elementos:
  - Carpeta del proyecto con el código fuente funcional.
  - Archivo `schema.sql` con el DDL real.
  - Archivo `analisis.pdf` con respuestas a preguntas de reflexión.
  - Capturas de pantalla demostrando las funcionalidades (crear, ver, editar, eliminar).

Parte 1 – Estructura de Base de Datos

1. **Diagrama entidad-relación completo**, incluyendo:
   - Todas las tablas.
   - Cardinalidades entre entidades.
   - Claves primarias, foráneas y atributos relevantes.
2. **Script DDL** (`schema.sql`) generado desde el ORM:
   - Definición completa de tablas.
   - Restricciones: `NOT NULL`, `UNIQUE`, `CHECK`, claves primarias y foráneas.
   - Tipos personalizados (mínimo 2).
3. **Script de inserción de datos** (`data.sql`):
   - Al menos 30 registros significativos.
   - Deben demostrar relaciones múltiples (por ejemplo, un estudiante con 2 o más clases).

Parte 2 – Lógica de Aplicación con ORM

- CRUD completo para la tabla principal usando exclusivamente el ORM.
- El mismo formulario debe permitir crear/asociar múltiples registros relacionados a través de una tabla intermedia.
- Validaciones deben existir en ambos niveles: aplicación y base de datos.
- Uso de al menos 2 tipos de datos personalizados.
- Vista (VIEW): Se debe crear una vista en la base de datos que combine la tabla principal, la tabla intermedia y la tabla relacionada para facilitar la visualización.
    - Esta vista será usada para poblar el índice en la interfaz.
    - No se permite hacer SELECTs directos desde múltiples tablas en el índice.

Parte 3 – Análisis (`analisis.pdf`)

Responde las siguientes preguntas:

1. ¿Por qué eligieron ese ORM y qué beneficios o dificultades encontraron?
2. ¿Cómo implementaron la lógica master-detail dentro del mismo formulario?
3. ¿Qué validaciones implementaron en la base de datos y cuáles en el código?
4. ¿Qué beneficios encontraron al usar tipos de datos personalizados?
5. ¿Qué ventajas ofrece usar una VIEW como base del índice en vez de una consulta directa?
6. ¿Qué escenarios podrían romper la lógica actual si no existieran las restricciones?
7. ¿Qué aprendieron sobre la separación entre lógica de aplicación y lógica de persistencia?
8. ¿Cómo escalaría este diseño en una base de datos de gran tamaño?
9. ¿Consideran que este diseño es adecuado para una arquitectura con microservicios?
10. ¿Cómo reutilizarían la vista en otros contextos como reportes o APIs?
11. ¿Qué decisiones tomaron para estructurar su modelo de datos y por qué?
12. ¿Cómo documentaron su modelo para facilitar su comprensión por otros desarrolladores?
13. ¿Cómo evitaron la duplicación de registros o errores de asignación en la tabla intermedia?

Rúbrica de Evaluación (100 puntos)

Criterio	Puntos
Diagrama E-R con cardinalidades	10
Script DDL correcto y completo	10
Inserción de datos significativos	10
CRUD funcional con escritura en varias tablas	15
Uso correcto de ORM (sin SQL manual)	10
Vista (VIEW) para el índice, bien implementada	10
Tipos de datos personalizados y validaciones	5
Análisis técnico en `analisis.pdf`	20
Capturas de evidencia funcional	10
Total	**100**