Campus:

Santa Fe

Materia:

Construcción de software y toma de decisiones

Nombre de la actividad:

5. Normalización de la base de datos del reto apoyándose en recursos de IA generativa

Nombre y Matrícula:

Emiliano Deyta Illescas A01785881

Amilka Daniela López Aguilar A01029277

Jin Sik Yoon A01026630

Profesores:

Octavio Navarro Hinojosa
Esteban Castillo Juarez
Gilberto Echeverría Furió

Grupo:

401

Fecha:

11/04/2024

Elementos del esquema normalizados con base en las clases

Separación de entidades: descomposición conceptual

Una de las prácticas centrales promovidas en el curso es la separación de cada entidad real del dominio en su propia tabla. Esto se basa en la Primera Forma Normal (1FN), que indica que cada atributo debe contener un solo valor por fila, y que las entidades deben estar claramente separadas para evitar datos multivaluados o dependencias implícitas.

En el esquema diseñado para *The Lost Sentinel*, cada entidad conceptual (como Jugador, Cuarto, Mapa, Objeto, NPC, Enemigo, Cofre, Tienda, etc.) se encuentra correctamente aislada en su propia tabla, cada una con su propia clave primaria. Esta estructura permite representar con claridad las relaciones del sistema sin redundancia y facilita su evolución.

La separación de las estadísticas (Estadisticas) del jugador de la entidad principal (Jugador) es un ejemplo claro de cumplimiento de la Segunda Forma Normal (2FN), ya que los datos dependientes funcionalmente de la clave (id_jugador) fueron extraídos a una tabla independiente. Esto evita almacenar estadísticas repetidas cada vez que se actualiza un jugador.

Del mismo modo, la entidad Inventario se separa como una tabla intermedia entre Jugador y Objeto, cumpliendo también con la 2FN y permitiendo modelar la relación N:M entre jugadores y los objetos que poseen.

El uso de Producto_Tienda como tabla intermedia entre Tienda y Objeto, y de Contenido_Cofre entre Cofre y Objeto, permite modelar relaciones muchos a muchos sin duplicar datos, lo cual sigue lo que se observa en los esquemas del curso, como film_actor o rental en Sakila.

Relaciones claras mediante claves foráneas

Todas las relaciones establecidas entre tablas del modelo utilizan claves foráneas bien definidas. Estas relaciones permiten mantener la integridad referencial, principio que fue enfatizado constantemente en las presentaciones del curso, particularmente en los ejemplos sobre PostgreSQL y la base de datos Sakila.

Por ejemplo:

- Jugador.id_cuarto hace referencia a Cuarto.id_cuarto, lo cual define en qué cuarto se encuentra un jugador.
- NPC.id_cuarto, Enemigo.id_cuarto y Jefe.id_cuarto también hacen referencia al mismo campo en Cuarto, permitiendo ubicar a cada entidad interactiva dentro del mapa.

- Estadisticas incluye únicamente el id_jugador como clave primaria y foránea, lo cual la liga directamente al jugador al que pertenece sin ambigüedad.

Este diseño evita que puedan existir registros que hagan referencia a datos que no existen (por ejemplo, estadísticas de un jugador que ha sido eliminado), lo cual se refuerza con la presencia de restricciones ON DELETE CASCADE en las relaciones más críticas.

Eliminación de dependencias transitivas (3FN)

Otra mejora basada en las presentaciones fue la eliminación de dependencias transitivas entre columnas no clave. Por ejemplo, no se almacena el nombre del mapa dentro de la tabla Cuarto, ya que dicha información ya se encuentra en Mapa, y se accede mediante la relación definida por id_mapa.

Del mismo modo, Jugador no contiene datos sobre las estadísticas de combate, ya que estos se encuentran centralizados en la tabla Estadisticas, lo cual permite que cada conjunto de estadísticas esté directamente relacionado con un jugador, sin duplicación ni ambigüedad.

Claves compuestas evitadas mediante surrogate keys y claves únicas

Aunque en los ejemplos del curso se presentan casos con claves primarias compuestas (como film_actor(film_id, actor_id)), en este modelo se optó por el uso de claves artificiales (AUTO_INCREMENT) y la definición de restricciones UNIQUE en los campos que lo requieren.

Por ejemplo:

- En Jugador, aunque usuario es un campo único por definición del dominio, se utiliza una clave primaria artificial id_jugador y se declara usuario como UNIQUE KEY.
- Lo mismo ocurre en Objeto, NPC y Enemigo, donde nombre es único, pero no clave primaria.

Esta práctica está alineada con la flexibilidad y el diseño escalable de bases de datos modernas.

Cambios sugeridos por IA

Herencia explícita entre Arma y Objeto

El diseño inicial incluía una tabla Arma con su propia clave primaria id_arma, que no tenía relación directa con Objeto. Esto generaba un problema de integridad referencial, ya que Objeto es el tipo base del cual se espera que Arma herede.

Se aplicó una solución común en bases de datos relacionales: utilizar el mismo identificador (id_objeto) como clave primaria y foránea en Arma, vinculándola directamente a Objeto. Esto simula una relación de herencia, donde Armaes un subtipo de Objeto y se garantiza que no exista un arma sin su correspondiente objeto padre.

Este patrón no se aborda en profundidad en las presentaciones, pero es ampliamente utilizado para representar modelos orientados a objetos dentro de bases de datos relacionales.

Uso de restricciones CHECK para validación de datos

Para asegurar la validez de los datos insertados, se agregaron cláusulas CHECK a varios campos. Si bien en las presentaciones no se insiste en este tipo de validaciones, su utilidad en proyectos reales es indiscutible.

Ejemplos implementados:

- dano INT CHECK (dano >= 0) en Arma, Jefe y Enemigo
- tipo TEXT CHECK (tipo IN ('espada', 'arco', 'barrita_magica', 'bomba')) en Arma
- stackable BOOLEAN en Objeto
- monedas, cantidad, vida_actual, etc., con restricciones de ser mayores o iguales a 0.

Estas reglas permiten evitar errores que podrían ocurrir desde el frontend o durante migraciones y pruebas.

Homogeneización de nombres de campos

Se decidió estandarizar el uso de nombres de campos para mantener compatibilidad internacional y evitar problemas de codificación. Por ejemplo, se reemplazó daño por dano, evitando el uso de la letra ñ, que puede causar errores en sistemas que no estén correctamente configurados para UTF-8.

Este cambio mejora la interoperabilidad y evita errores al realizar consultas desde herramientas externas, scripts u ORMs.

Campos de auditoría para trazabilidad

Aunque no es un requerimiento del curso, se agregaron campos fecha_creacion DATETIME DEFAULT CURRENT_TIMESTAMP en varias tablas clave (Mapa, Jugador, Cofre, etc.). Esto permite llevar un seguimiento del momento en que se crean los registros sin requerir lógica adicional en la aplicación.

Esto también es útil para depuración, estadísticas del sistema, o auditoría de comportamiento.

Aplicación sistemática de ON DELETE y ON UPDATE

Se definieron reglas explícitas para controlar qué ocurre cuando se elimina o actualiza una fila referenciada. Estas reglas incluyen:

- ON DELETE CASCADE en tablas como Inventario, Estadisticas, Contenido_Cofre, Producto_Tienda, para eliminar en cascada cuando se borra el padre.
- ON DELETE SET NULL en algunos casos como Jugador.id_cuarto, para conservar al jugador aunque se elimine el cuarto.

Estas decisiones buscan balancear la limpieza automática con la preservación de datos según el contexto de la entidad.

Adición de índices para optimizar búsquedas

Para mejorar el rendimiento en consultas frecuentes, se agregaron índices manuales, como en el campo nombre de Objeto:

CREATE INDEX idx_nombre_objeto ON Objeto(nombre);

Esto es útil para acelerar búsquedas por nombre o filtros en interfaces, algo que será relevante si la base crece o si se conecta a un sistema de comercio o inventario en tiempo real.

Importancia de normalización y lenguaje generativo

La normalización es una técnica esencial en el diseño de bases de datos relacionales. Su objetivo principal es eliminar redundancia, evitar inconsistencias y mejorar la integridad de los datos. A través de formas normales (1FN, 2FN, 3FN), se logra dividir la información en tablas específicas, conectadas mediante claves foráneas, lo que facilita la mantenibilidad y escalabilidad del sistema.

Durante el desarrollo del esquema para *The Lost Sentinel*, la normalización permitió estructurar correctamente entidades como Jugador, Inventario, Objeto y Tienda, evitando duplicación de información y facilitando las relaciones entre ellas. Gracias a esto, el sistema resultó más limpio, ordenado y preparado para crecer sin errores lógicos.

Además, se contó con el apoyo de herramientas de lenguaje generativo como modelos de IA, que ayudaron a revisar errores, sugerir mejores prácticas (como el uso de CHECK, ON DELETE CASCADE o el modelado de herencia con claves foráneas) y acelerar el diseño técnico. Sin embargo, estas herramientas no

reemplazaron el trabajo del estudiante desarrollador: el análisis del problema, la interpretación del diagrama y la toma de decisiones siempre dependieron del criterio humano.