Diseño de Bases de Datos – Material complementario

1. Metodologías de Diseño de Bases de Datos: Modelo Entidad-Relación

El diseño de una base de datos no es un proceso improvisado; se basa en metodologías estructuradas. La más utilizada para el diseño conceptual es el **Modelo Entidad-Relación (MER)**. Este modelo permite representar la estructura de los datos de forma abstracta y comprensible antes de pasar a un diseño técnico.

- **Entidades:** Son objetos o conceptos del mundo real que tienen existencia independiente y sobre los que se desea almacenar información. En el MER, se representan con un rectángulo.
    - **Ejemplo:** En el trabajo práctico, las entidades podrían ser CLIENTE, LIBRO, EMPLEADO y VENTA.
- **Atributos:** Son las propiedades o características que describen a una entidad. Se representan con una elipse y se asocian a una entidad.
    - **Ejemplo:** La entidad CLIENTE podría tener atributos como Nombre_Cliente, Apellido_Cliente y Email_Cliente.
- **Relaciones:** Son las asociaciones entre dos o más entidades. Se representan con un rombo. Cada relación tiene una cardinalidad que describe cuántas instancias de una entidad se asocian con las instancias de otra.
    - **Tipos de Cardinalidad:**
        - **Uno a Uno (1:1):** Una instancia de la entidad A se relaciona con una única instancia de la entidad B.
        - **Uno a Muchos (1:N):** Una instancia de la entidad A se relaciona con muchas instancias de la entidad B.
        - **Muchos a Muchos (N:M):** Varias instancias de la entidad A se relacionan con varias instancias de la entidad B.
    - **Ejemplo:** La relación entre CLIENTE y VENTA es de **Uno a Muchos**, ya que un cliente puede realizar varias ventas. La relación entre VENTA y LIBRO es de **Muchos a Muchos**, ya que una venta puede incluir muchos libros y un libro puede estar en muchas ventas.

2. Fases del Diseño de Bases de Datos

El diseño de una base de datos se divide en tres fases principales que permiten una transición organizada desde un concepto de alto nivel hasta la implementación física.

- **1. Diseño Conceptual:** Es la primera fase y se enfoca en capturar los requerimientos de datos del negocio. Utiliza el Modelo Entidad-Relación (MER) para crear una representación de los datos sin considerar las limitaciones técnicas. El resultado es un esquema de alto nivel que es fácil de entender para los usuarios no técnicos. En esta etapa se identifican las entidades, sus atributos y las relaciones.

- **2. Diseño Lógico:** En esta fase, el modelo conceptual se traduce a un esquema que puede ser implementado en un **Sistema de Gestión de Bases de Datos (SGBD)** relacional. Se crean las tablas, se definen las claves primarias (identificadores únicos) y las claves foráneas (para mantener las relaciones), y se aplica el proceso de normalización para eliminar redundancias. Este es el punto en el que se trabaja con las Formas Normales.

- **3. Diseño Físico:** Es la fase de implementación. El diseño lógico se traduce a la estructura física de la base de datos, considerando el SGBD específico que se va a utilizar (por ejemplo, MySQL, PostgreSQL o SQL Server). Se definen tipos de datos específicos, índices para optimizar el rendimiento de las consultas y otras configuraciones técnicas que son invisibles para los usuarios finales.

3. Proceso de Normalización

La normalización es un proceso sistemático para reorganizar las tablas de una base de datos relacional para reducir la redundancia de datos y mejorar la integridad de los mismos. El trabajo práctico te pide aplicar las tres primeras formas normales.

Primera Forma Normal (1FN)

Una tabla está en 1FN si:

- Cada atributo contiene valores atómicos (indivisibles).
- No hay grupos de atributos repetitivos.
- **Violaciones en el trabajo práctico:**
    - La tabla

VENTAS_LIBRERIA tiene valores atómicos en la mayoría de sus columnas, pero la combinación de ID_Venta y ID_Libro parece indicar que una venta puede tener múltiples ítems. La fila con

ID_Venta 101, por ejemplo, está duplicada para cada libro, lo que sugiere un grupo repetitivo de datos de libros dentro de una misma venta. Esto es una violación.

- **Solución para 1FN:** Para resolver esta violación, se debe identificar una clave primaria compuesta que haga que cada fila sea única. La clave primaria de la nueva tabla será la combinación de

ID_Venta e ID_Libro. Esto asegura que cada fila representa un ítem único vendido en una transacción.

- **Tabla Resultante (en 1FN):** La tabla se vería muy similar a la inicial, pero se entenderá que la clave primaria es la combinación de ID_Venta e ID_Libro.

Segunda Forma Normal (2FN)

Una tabla está en 2FN si cumple con 1FN y todos los atributos no clave dependen completamente de la clave primaria completa. Si la clave primaria es compuesta, no puede haber atributos que dependan de solo una parte de ella.

- **Violaciones en el trabajo práctico:**
    - La clave primaria de la tabla en 1FN es (ID_Venta, ID_Libro).
    - Los atributos del cliente (

Nombre_Cliente, Apellido_Cliente, Email_Cliente, etc.) solo dependen de ID_Venta, no de ID_Libro.

    - Los atributos del libro (

Título_Libro, Autor_Libro, Genero_Libro, etc.) solo dependen de ID_Libro, no de ID_Venta.

    - Los atributos del empleado (

Nombre_Empleado, DNI_Empleado) solo dependen de ID_Venta.

    - Todas estas son **dependencias parciales**.
- **Solución para 2FN:** Se dividen las tablas para eliminar estas dependencias parciales. Se crean nuevas tablas con atributos que dependen solo de una parte de la clave original.
    - Tabla VENTAS con clave primaria ID_Venta.
    - Tabla CLIENTES con clave primaria Email_Cliente (o un nuevo ID_Cliente).
    - Tabla LIBROS con clave primaria ID_Libro.
    - Tabla EMPLEADOS con clave primaria DNI_Empleado (o un nuevo ID_Empleado).
    - Una tabla intermedia (VENTAS_DETALLE) con clave primaria (ID_Venta, ID_Libro) para registrar la Cantidad y el Precio_Unitario de cada libro en una venta.
    - Se establecen claves foráneas para mantener las relaciones.

Tercera Forma Normal (3FN)

Una tabla está en 3FN si cumple con 2FN y no tiene dependencias transitivas. Una dependencia transitiva ocurre cuando un atributo no clave depende de otro atributo no clave, en lugar de depender directamente de la clave primaria.

- **Violaciones en el trabajo práctico:**
    - Después de la 2FN, la tabla VENTAS podría incluir atributos del cliente o empleado que no se separaron correctamente. El

Nombre_Empleado y DNI_Empleado dependen del ID_Venta, pero el nombre del empleado (Nombre_Empleado) depende del DNI del empleado (DNI_Empleado). Es una dependencia transitiva.

    - De forma similar, los datos del cliente (Nombre_Cliente, Apellido_Cliente, Tel_Cliente, Dirección_Cliente, Ciudad_Cliente) dependen de Email_Cliente, y no directamente de la clave primaria de la tabla de ventas, que podría ser ID_Venta. Esto requiere una tabla separada para CLIENTES.

- **Solución para 3FN:** Se dividen las tablas para eliminar estas dependencias transitivas.
    - Tabla EMPLEADOS con DNI_Empleado como clave primaria.
    - Tabla CLIENTES con Email_Cliente o un nuevo ID como clave primaria.
    - La tabla VENTAS contendrá claves foráneas que apunten a DNI_Empleado y al ID del cliente.

La aplicación de estas formas normales da como resultado un esquema de base de datos robusto con menos redundancia de datos y una integridad referencial mejorada, lo que es esencial para el sistema de ventas que se pide en el trabajo práctico.