NORMALIZACIÓN DE BASE DE DATOS

La **normalización** es un proceso en el diseño de bases de datos que se utiliza para organizar los datos de manera eficiente. Su objetivo principal es **reducir la redundancia de los datos** y mejorar la integridad de los mismos. A través de la normalización, los datos se estructuran en tablas y relaciones de manera que se minimicen las **anomalías durante las operaciones de inserción, actualización y eliminación**.

La normalización se divide en varias **formas normales** (FN), que son reglas que deben seguir las tablas de la base de datos. A continuación, te explico las formas normales más comunes:

**1. Primera Forma Normal (1FN):**

* Una tabla está en primera forma normal si todos sus atributos contienen valores atómicos (indivisibles), es decir, no debe haber conjuntos de valores o listas dentro de una columna.
* **Ejemplo**: Si en una tabla de "Clientes" tienes una columna que almacena múltiples números de teléfono en una celda, se debe dividir esta información en filas separadas o en otra tabla relacionada.

**2. Segunda Forma Normal (2FN):**

* Una tabla está en segunda forma normal si cumple con la 1FN y, además, todos los atributos que no son claves dependen completamente de la clave primaria.
* **Evita la dependencia parcial**: Si una tabla tiene una clave primaria compuesta (formada por varios campos), ningún atributo que no sea clave debe depender solo de parte de la clave primaria.
* **Ejemplo**: En una tabla de "Pedidos", si el identificador del cliente y el producto se usan como clave primaria, todos los campos de la tabla deben depender tanto del cliente como del producto. Si solo dependen del cliente o del producto, deben estar en otra tabla.

**3. Tercera Forma Normal (3FN):**

* Una tabla está en tercera forma normal si cumple con la 2FN y, además, no tiene dependencias transitivas.
* **Evita la dependencia transitiva**: Los atributos no clave deben depender únicamente de la clave primaria y no de otros atributos no clave.
* **Ejemplo**: En una tabla de "Empleados", si tienes el campo "Departamento" y "Nombre del Jefe", y "Nombre del Jefe" depende del "Departamento", entonces "Nombre del Jefe" no debería estar en la tabla de "Empleados", sino en una tabla separada de "Departamentos".

**Beneficios de la normalización:**

1. **Reducción de la redundancia**: Al eliminar datos duplicados, se evita almacenar la misma información en múltiples lugares.
2. **Mejora de la integridad de los datos**: Ayuda a mantener los datos consistentes y a prevenir anomalías durante las operaciones de actualización, eliminación o inserción.
3. **Eficiencia en las consultas**: Las tablas más pequeñas y bien estructuradas pueden acelerar las consultas de bases de datos.

**Desventajas:**

1. **Mayor complejidad**: En algunos casos, el proceso de normalización puede generar muchas tablas y relaciones, lo que puede complicar la consulta y el diseño.
2. **Rendimiento**: En bases de datos muy grandes, a veces las tablas muy normalizadas requieren muchas uniones (joins), lo que puede afectar el rendimiento de las consultas.

En resumen, la normalización es clave para crear bases de datos que sean eficientes, libres de redundancias y que mantengan la integridad de los datos.

DEPENDENCIAS FUNCIONALES

Las **dependencias funcionales** son un concepto fundamental en el diseño y normalización de bases de datos relacionales. Se utilizan para identificar y formalizar las relaciones entre atributos dentro de una tabla. En esencia, una dependencia funcional describe cómo los valores de ciertos atributos de una tabla determinan los valores de otros atributos.

**Definición de Dependencia Funcional:**

Se dice que existe una **dependencia funcional** entre dos conjuntos de atributos en una tabla cuando el valor de un conjunto de atributos (A) determina de manera única el valor de otro conjunto de atributos (B).

* Formalmente:  
  Un atributo o conjunto de atributos **A** determina un atributo o conjunto de atributos **B** si para cualquier par de tuplas en la tabla que tengan el mismo valor en A, también tendrán el mismo valor en B.
  + Se denota:  
    **A → B**
    - Lo que significa que "A determina a B" o "B depende funcionalmente de A".

**Ejemplo básico:**

Imagina una tabla **Estudiantes(ID\_Estudiante, Nombre, Dirección, Teléfono)**.

* En este caso, el **ID\_Estudiante** determina de manera única los valores de **Nombre**, **Dirección** y **Teléfono**.
* Esto se puede expresar como:
  + **ID\_Estudiante → Nombre, Dirección, Teléfono**
  + Lo que significa que si conoces el **ID\_Estudiante**, puedes determinar de forma única el **Nombre**, **Dirección**, y **Teléfono** de ese estudiante.

**Tipos de Dependencias Funcionales:**

1. **Dependencia funcional completa**:
   * Un atributo B depende funcionalmente de un conjunto de atributos A de forma completa si no hay una parte de A que por sí sola pueda determinar a B.
   * **Ejemplo**: Si tienes una tabla **Pedidos(ID\_Pedido, ID\_Producto, Cantidad, Precio)**, el **Precio** depende completamente del **ID\_Producto**, no solo de **ID\_Pedido**.
   * **Formalmente**: ID\_Producto → Precio, lo que significa que para un ID de producto, el precio es único.
2. **Dependencia funcional parcial**:
   * Una dependencia funcional parcial ocurre cuando un atributo B depende de una parte de un conjunto de atributos A, en lugar de depender del conjunto completo.
   * **Ejemplo**: Si tienes una tabla **Pedidos(ID\_Pedido, ID\_Producto, Cantidad, Precio\_Total)**, y el precio total depende del **ID\_Producto**, pero no del **ID\_Pedido** (es decir, solo depende de una parte de la clave compuesta). En este caso, hay una dependencia funcional parcial.
   * **Formalmente**: ID\_Producto → Precio\_Total (no necesita del ID\_Pedido).
3. **Dependencia funcional transitiva**:
   * Una dependencia funcional transitiva ocurre cuando un atributo A determina un atributo B, y B, a su vez, determina un atributo C.
   * **Ejemplo**: Si tienes una tabla **Empleados(ID\_Empleado, ID\_Departamento, Nombre\_Departamento, Nombre\_Jefe)**, el **ID\_Departamento** determina el **Nombre\_Jefe**. A su vez, el **ID\_Empleado** determina el **ID\_Departamento**, lo que hace que haya una dependencia transitiva de **ID\_Empleado → Nombre\_Jefe**.
   * **Formalmente**: ID\_Empleado → ID\_Departamento y ID\_Departamento → Nombre\_Jefe. Por lo tanto, se puede decir que existe una dependencia transitiva: **ID\_Empleado → Nombre\_Jefe**.
4. **Dependencia funcional trivial**:
   * Una dependencia funcional trivial ocurre cuando un atributo o conjunto de atributos determina un subconjunto de sí mismo.
   * **Ejemplo**: En la tabla **Estudiantes(ID\_Estudiante, Nombre)**, la dependencia **ID\_Estudiante → ID\_Estudiante** es trivial porque un atributo siempre determina su propio valor.
   * **Formalmente**: A → A.

**Importancia de las Dependencias Funcionales en la Normalización:**

Las dependencias funcionales son clave para el proceso de **normalización** en el modelado de bases de datos, ya que ayudan a identificar y eliminar redundancias y posibles anomalías. Aquí te explico cómo:

1. **Primera Forma Normal (1FN)**:
   * La dependencia funcional no juega un papel directo en esta forma, pero las tablas deben cumplir con la 1FN para que las dependencias funcionales puedan ser evaluadas.
2. **Segunda Forma Normal (2FN)**:
   * Una tabla está en 2FN si no tiene dependencias funcionales parciales. Es decir, si un atributo no clave depende de parte de una clave compuesta, debe eliminarse de la tabla y colocarse en una nueva tabla para eliminar la dependencia parcial.
3. **Tercera Forma Normal (3FN)**:
   * Para que una tabla esté en 3FN, no debe tener dependencias funcionales transitivas. Esto significa que los atributos no clave no deben depender de otros atributos no clave. Se deben eliminar estas dependencias para evitar que los cambios en un atributo afecten a otros de manera no deseada.

**Ejemplo práctico de normalización usando dependencias funcionales:**

Supongamos que tenemos la siguiente tabla no normalizada:

| **ID\_Empleado** | **Nombre\_Empleado** | **ID\_Departamento** | **Nombre\_Departamento** | **Jefe\_Departamento** |
| --- | --- | --- | --- | --- |
| 1 | María | D1 | Ventas | Carlos |
| 2 | Juan | D1 | Ventas | Carlos |
| 3 | Pedro | D2 | Finanzas | Ana |

**Dependencias funcionales detectadas:**

* **ID\_Empleado → Nombre\_Empleado, ID\_Departamento** (ID\_Empleado determina al empleado y su departamento).
* **ID\_Departamento → Nombre\_Departamento, Jefe\_Departamento** (ID\_Departamento determina el nombre del departamento y su jefe).

**Proceso de normalización (Hasta 3FN):**

1. **Eliminar la dependencia transitiva**:
   * Hay una dependencia transitiva de **ID\_Empleado → Jefe\_Departamento**, porque **ID\_Empleado → ID\_Departamento** y **ID\_Departamento → Jefe\_Departamento**. Por lo tanto, **ID\_Empleado → Jefe\_Departamento** es una dependencia funcional transitiva.
2. **Dividir la tabla**:
   * Crearemos dos tablas para eliminar esta dependencia:

**Tabla Empleados**:

| **ID\_Empleado** | **Nombre\_Empleado** | **ID\_Departamento** |
| --- | --- | --- |
| 1 | María | D1 |
| 2 | Juan | D1 |
| 3 | Pedro | D2 |

**Tabla Departamentos**:

| **ID\_Departamento** | **Nombre\_Departamento** | **Jefe\_Departamento** |
| --- | --- | --- |
| D1 | Ventas | Carlos |
| D2 | Finanzas | Ana |

De este modo, hemos eliminado la dependencia transitiva y ahora las tablas cumplen con la **3FN**.

**Conclusión:**

Las **dependencias funcionales** son fundamentales para determinar cómo se estructuran los datos en una base de datos relacional. Ayudan a eliminar redundancias, asegurar la integridad de los datos y garantizar que las bases de datos sean eficientes y libres de anomalías. Al comprender y gestionar adecuadamente las dependencias funcionales, se puede lograr un diseño de base de datos más sólido y escalable.