1. DISEÑO DE UNA BASE DE DATOS

1.1 Etapas del diseño de bases de datos

En la era de la digitalización, los datos son un activo valioso. Las bases de datos son el corazón que bombea este activo a través de la organización, permitiéndole operar, analizar y tomar decisiones. Para los recién llegados a este campo, el diseño de una base de datos puede parecer desalentador. Sin embargo, este proceso se puede desglosar en etapas manejables que facilitan la creación de una base de datos que es eficiente, eficaz y capaz de satisfacer las necesidades de la organización. Aquí están las etapas principales:

1) Requisitos de Recolección: Imagina que estás construyendo una casa. Antes de empezar a construir, necesitas saber cuántas habitaciones debería tener, si necesita un garaje, qué tipo de cocina quieres, etc. De manera similar, antes de empezar a diseñar una base de datos, necesitas entender qué se espera de ella. ¿Qué tipo de información almacenará? ¿Cómo se utilizará la información? ¿Quiénes serán los usuarios de la base de datos? Estas son algunas de las preguntas que se deben responder en esta etapa. Los requisitos se recogen a través de entrevistas, cuestionarios, y observando los procesos de negocio existentes.

2) Diseño Conceptual: Una vez que tienes claro lo que se necesita, el siguiente paso es crear un esquema conceptual de cómo se relacionarán los diferentes tipos de información entre sí. Esto es similar a un arquitecto dibujando el plano de una casa. El diseño conceptual de una base de datos podría incluir, por ejemplo, que cada Pedido estará vinculado a un Cliente y a uno o varios Productos.

3) Diseño Lógico: En esta etapa, el esquema conceptual se traduce en un diseño más detallado. Aquí es donde decides cuáles serán las tablas en la base de datos, qué datos contendrá cada tabla, y cómo se conectarán estas tablas entre sí. Por ejemplo, podrías tener una tabla para Clientes, una tabla para Pedidos, y una tabla para Productos. Cada Pedido estaría vinculado a un Cliente y tendría múltiples entradas de Productos.

4) Diseño Físico: Después de tener un diseño lógico, se definen los aspectos técnicos de cómo se almacenará y accederá a la base de datos en el hardware. Esto puede incluir aspectos como el formato de almacenamiento de datos, cómo se indexarán los datos para un acceso rápido, y qué medidas de seguridad se implementarán.

5) Implementación: Ahora es el momento de construir la base de datos utilizando un Sistema de Gestión de Bases de Datos (DBMS). Aquí es donde se crean las tablas, se establecen las relaciones, y se llenan las tablas con datos.

6) Pruebas: Al igual que querrías inspeccionar una casa antes de mudarte, querrás probar la base de datos antes de usarla en un entorno de producción. ¿Funciona como debería? ¿Se puede acceder a los datos de manera eficiente? ¿Es segura?

7) Mantenimiento y Evaluación: Después de que la base de datos esté en funcionamiento, es importante supervisarla y ajustarla de acuerdo a las necesidades cambiantes del negocio. Al igual que una casa necesita mantenimiento regular y reparaciones ocasionales, una base de datos también requiere atención continua para garantizar su rendimiento y eficiencia. Esto podría implicar ajustar

la forma en que se indexan los datos, añadir nuevas tablas o campos según sea necesario, y garantizar que la base de datos siga siendo segura a medida que surgen nuevas amenazas.

Para ilustrar estas etapas, consideremos un ejemplo de una base de datos para una librería:

1. **Requisitos de Recolección**: Necesitamos almacenar información sobre los libros, los clientes, y las compras que realizan. Queremos poder buscar libros por título, autor o género, y queremos poder ver el historial de compras de un cliente.
2. **Diseño Conceptual**: Decidimos que tendremos entidades principales para los Libros, los Clientes y las Compras. Cada Libro tendrá un título, un autor y un género. Cada Cliente tendrá un nombre y una dirección de correo electrónico. Cada Compra estará vinculada a un Cliente enlace a un Cliente y enlace a uno o varios Libros.
3. **Diseño Lógico**: Decidimos tener una tabla para Libros, una para Clientes, y una para Compras. También tendremos una tabla de Detalles de Compra para almacenar la información sobre qué libros se compraron en cada compra.
4. **Diseño Lógico Físico**: Decidimos que usaremos MySQL como nuestro DBMS. También decidimos utilizar el almacenamiento InnoDB para nuestras tablas, ya que ofrece una buena combinación de rendimiento y características de seguridad.
5. **Implementación**: Creamos nuestras tablas en MySQL y las llenamos con datos.
6. **Pruebas:** Probamos nuestra base de datos realizando varias consultas para buscar libros y compras. También probamos nuestras medidas de seguridad asegurándonos de que sólo los usuarios autorizados puedan acceder a la base de datos.
7. **Mantenimiento y Evaluación:** Mantenemos nuestra base de datos monitoreando su rendimiento y asegurándose de que se mantenga segura. También realizamos cambios en la base de datos si es necesario para adaptarse a las cambiantes necesidades de nuestro negocio.

En resumen, el diseño de una base de datos es un proceso complejo, pero manejable, que implica la recopilación de requisitos, la creación de un diseño conceptual, la realización de un diseño lógico y físico, la implementación de la base de datos, las pruebas y el mantenimiento y la evaluación continua. Al seguir estas etapas, puedes crear una base de datos que sea eficiente, eficaz y capaz de satisfacer las necesidades de tu organización o negocio.

1.2 Teoría de la Normalización

La normalización es un conjunto de reglas y procesos que se siguen para diseñar una base de datos eficiente y con una estructura sólida. Ayuda a eliminar la redundancia de datos y garantiza la integridad de los datos. Estas reglas se conocen como "formas normales", y hay varias formas normales que se aplican en una secuencia específica para normalizar una base de datos. Aquí hay una descripción detallada de cada forma normal con ejemplos para ayudar a entender mejor el proceso de normalización:

1. Primera Forma Normal (1NF)

En la 1NF, cada tabla debe tener claves primarias, que son identificadores únicos para cada fila de la tabla. Además, cada columna en una tabla debe tener un valor único y atómico (indivisible). No se permiten grupos repetitivos de datos.

Ejemplo: Imagina una tabla "Clientes" que contiene información sobre los clientes y sus números de teléfono. Si un cliente tiene varios números de teléfono, no puedes guardar todos los números en una única columna. En su lugar, debes dividir los números de teléfono en varias filas, cada una con su propia clave primaria.

2. Segunda Forma Normal (2NF)

La 2NF se aplica a las bases de datos que ya están en 1NF. En 2NF, todas las columnas no clave deben depender completamente de la clave primaria. Es decir, no puede haber dependencia parcial de la clave primaria.

Ejemplo: Supongamos que tienes una tabla "Pedidos" que incluye información sobre los productos pedidos y sus precios. Si la clave primaria de la tabla es una combinación de "ID de Pedido" e "ID de Producto", el precio del producto no debe estar en la tabla "Pedidos", ya que depende solo del "ID de Producto". En cambio, el precio del producto debe estar en una tabla separada, como "Productos".

3. Tercera Forma Normal (3NF)

La 3NF se aplica a las bases de datos que ya están en 2NF. En 3NF, todas las columnas deben depender directamente de la clave primaria. No pueden existir dependencias transitivas.

Ejemplo: Imagina una tabla "Empleados" que incluye información sobre los empleados y sus departamentos. Si la clave primaria de la tabla es "ID de Empleado", la columna "Nombre del Departamento" no debe estar en la tabla "Empleados", ya que depende del "ID del Departamento" y no directamente del "ID del Empleado". En su lugar, el "Nombre del Departamento" debe estar en una tabla separada, como "Departamentos".

- Normalización de bases de datos: https://www.youtube.com/watch?v=fxbC4cwnb1U&t=1s

4. Forma Normal de Boyce-Codd (BCNF)

La BCNF es una extensión de 3NF que maneja ciertos casos especiales donde 3NF podría permitir redundancias. En BCNF, cada determinante (un conjunto de atributos que determina otro conjunto de atributos) debe ser una superclave (un conjunto mínimo de atributos que puede identificar de manera única una fila en la tabla).

Ejemplo: Supongamos que tienes una tabla "Cursos" con información sobre cursos y profesores. Si tanto el "ID del Curso" como el "ID del Profesor" pueden determinar el "Horario del Curso", estos atributos deben ser tratados como superclaves. Por lo tanto, si solo el "ID del Curso" es la

clave primaria, la tabla no está en BCNF. Para cumplir con BCNF, debes dividir la tabla en dos: una tabla "Cursos" que asocia "ID del Curso" con "Horario del Curso" y una tabla "Profesores" que asocia "ID del Profesor" con "Horario del Curso".

5. Cuarta Forma Normal (4NF)

La 4NF se aplica a las bases de datos que ya están en BCNF y trata la cuestión de las dependencias multivaluadas. En 4NF, no puede haber dependencias multivaluadas. Una dependencia multivaluada ocurre cuando un atributo en una tabla puede llevar a múltiples valores de otro atributo.

Ejemplo: Supón que tienes una tabla "Estudiantes" que asocia a los estudiantes con los cursos que están tomando y los libros que requieren para esos cursos. Un estudiante puede estar tomando múltiples cursos y cada curso puede requerir múltiples libros. Aquí, tienes una dependencia multivaluada entre "Cursos" y "Libros". Para cumplir con 4NF, debes dividir esta tabla en dos: una tabla "Estudiantes_Cursos" que asocia a los estudiantes con sus cursos, y una tabla "Cursos_Libros" que asocia los cursos con los libros requeridos.

6. Quinta Forma Normal (5NF o Proyección-Join)

La 5NF, también conocida como Proyección-Join, trata casos especiales donde un diseño 4NF aún puede presentar redundancias. En 5NF, una tabla está en 5NF si cada dependencia de join es una dependencia de superclave. Un join es una operación que combina filas de dos o más tablas en función de una columna relacionada entre ellas.

Ejemplo: Considera una tabla "Proveedores" que almacena qué proveedor suministra qué parte a qué proyecto. Si un proveedor puede suministrar diferentes partes a diferentes proyectos, puedes tener redundancias si un proveedor suministra la misma parte a diferentes proyectos. Para cumplir con 5NF, puedes dividir esta tabla en tres: una tabla "Proveedores", una tabla "Partes" y una tabla "Proyectos", y puedes tener tablas separadas para las relaciones "Proveedores_Partes", "Partes_Proyectos" y "Proveedores_Proyectos".

- Video Normalización hasta 5FN: https://www.youtube.com/watch?v=pHkv7LxkLIo

En resumen, la normalización es un proceso esencial en el diseño de bases de datos que ayuda a minimizar la redundancia y a maximizar la integridad de los datos. Al entender y aplicar estas formas normales, puedes crear bases de datos que sean eficientes, escalables y fáciles de mantener.

1.3 Aplicación de la teoría de la normalización al diseño de bases de datos relacionales

Las bases de datos relacionales, que son el tipo de base de datos más comúnmente utilizado, se benefician enormemente de la normalización. Esta metodología, que se aplica principalmente

durante el diseño lógico de la base de datos, tiene como objetivo optimizar la estructura de las tablas de la base de datos para minimizar la redundancia y maximizar la integridad y coherencia de los datos.

Importancia de la normalización en las bases de datos relacionales

La normalización es esencial para cualquier base de datos relacional por varias razones. En primer lugar, ayuda a eliminar la redundancia de datos, lo que puede resultar en un desperdicio significativo de espacio de almacenamiento. Además, la redundancia de datos puede llevar a inconsistencias y problemas de mantenimiento, especialmente cuando se trata de actualizar, insertar o eliminar datos.

Además, la normalización mejora la integridad de los datos, que es la precisión y la consistencia de los datos almacenados en la base de datos. Con una base de datos bien normalizada, es menos probable que se produzcan errores y problemas, y los datos tienden a ser más confiables y precisos.

Aplicar la teoría de la normalización al diseño de bases de datos relacionales

Cuando se diseña una base de datos relacional, la normalización comienza con la Primera Forma Normal (1NF) y continúa a través de las formas normales subsiguientes hasta que se logra el nivel deseado de normalización, que puede ser hasta la Quinta Forma Normal (5NF) dependiendo de las necesidades específicas del sistema de base de datos.

Veamos un ejemplo práctico de cómo se aplica la teoría de la normalización al diseño de una base de datos relacional.

Ejemplo práctico:

Supongamos que estamos diseñando una base de datos para una librería. Tenemos una tabla inicial llamada "Ventas" que incluye los siguientes campos: ID_Venta, ID_Cliente, Nombre_Cliente, ID_Libro, Título_Libro, Autor_Libro, Precio_Libro.

1NF: En esta etapa, nos aseguramos de que cada columna tenga un valor atómico y que cada fila sea única. En este caso, ya cumplimos con 1NF.

2NF: Aquí, debemos asegurarnos de que todas las columnas no clave dependan completamente de la clave primaria. En nuestra tabla, los campos Nombre_Cliente, Título_Libro, Autor_Libro y Precio_Libro no dependen completamente de la clave primaria (ID_Venta). Entonces, dividimos la tabla en tres: "Ventas" (ID_Venta, ID_Cliente, ID_Libro), "Clientes" (ID_Cliente, Nombre_Cliente) y "Libros" (ID_Libro, Título_Libro, Autor_Libro, Precio_Libro).

3NF: En 3NF, todas las columnas deben depender directamente de la clave primaria. En nuestra tabla "Libros", el Precio_Libro depende del Título_Libro y del Autor_Libro, no directamente del ID_Libro. Por lo tanto, debemos dividir la tabla "Libros" en dos: "Libros" (ID_Libro, Título_Libro, Autor_Libro) y "Precios" (ID_Libro, Precio_Libro). Ahora, todas las tablas están en 3NF.

BCNF: En BCNF, cada determinante debe ser una superclave. En este caso, todas nuestras tablas ya están en BCNF, ya que todas las dependencias funcionales son sobre superclaves.

4NF: La 4NF trata las dependencias multivaluadas. Si un libro puede tener varios autores, tendríamos una dependencia multivaluada entre ID_Libro y Autor_Libro. Para resolver esto, dividiríamos la tabla "Libros" en dos tablas: "Libros" (ID_Libro, Título_Libro) y "Autores" (ID_Libro, Autor_Libro).

5NF: En 5NF, una tabla está en 5NF si cada dependencia de join es una dependencia de superclave. En este caso, todas nuestras tablas ya están en 5NF, ya que no hay dependencias de join que no sean dependencias de superclave.

A lo largo de este proceso, la aplicación de la teoría de la normalización nos ha permitido eliminar la redundancia y garantizar la integridad de los datos en nuestra base de datos relacional.

Es importante recordar que, aunque la normalización es un objetivo importante en el diseño de bases de datos, no siempre es necesario o práctico llegar hasta la 5NF. A veces, puede haber razones de rendimiento o de negocio para permitir cierto grado de denormalización. Sin embargo, entender y aplicar los principios de la normalización sigue siendo fundamental para el diseño eficaz de bases de datos relacionales.

1.4 Desnormalización de bases de datos

La desnormalización es un enfoque estratégico en el diseño de bases de datos que, a diferencia de la normalización, busca optimizar el rendimiento a través de ciertos grados de redundancia. Este proceso es una decisión intencionada y controlada que se toma después de la normalización. Aunque puede parecer contra intuitivo después de nuestro análisis detallado de la normalización y sus beneficios, la desnormalización tiene sus propias ventajas y puede ser una herramienta valiosa en ciertos casos.

¿Por qué desnormalizar?

La normalización es un proceso de diseño que favorece la integridad de los datos, pero a veces puede conducir a una estructura de base de datos que no es óptima desde el punto de vista del rendimiento. Esto se debe a que la normalización a menudo conduce a múltiples tablas relacionadas entre sí, lo que puede requerir operaciones de unión costosas durante las consultas.

La desnormalización busca mejorar el rendimiento de lectura de la base de datos al reducir la cantidad de tablas y, por lo tanto, el número de uniones necesarias para recuperar los datos. Esto puede hacer que las operaciones de lectura sean más rápidas y eficientes. Además, en algunos casos, la desnormalización puede simplificar las consultas, lo que puede facilitar el trabajo de los desarrolladores.

Es importante tener en cuenta que la desnormalización es un proceso que debe manejarse con cuidado. Aunque puede mejorar el rendimiento de lectura, también puede introducir redundancia y aumentar la complejidad de las operaciones de escritura. Por lo tanto, es vital equilibrar las necesidades de lectura y escritura, y considerar cuidadosamente las implicaciones de la desnormalización.

Ejemplo práctico:

Volviendo a nuestro ejemplo de la librería, supongamos que la aplicación más comúnmente utilizada requiere mostrar la información de ventas junto con el nombre del cliente, el título del libro y el precio del libro. En la estructura de base de datos normalizada, esta consulta requeriría múltiples uniones entre las tablas "Ventas", "Clientes", "Libros" y "Precios".

Para mejorar el rendimiento de lectura, podríamos desnormalizar nuestra base de datos combinando estas tablas en una sola tabla "Ventas" con los siguientes campos: ID_Venta, ID_Cliente, Nombre_Cliente, ID_Libro, Título_Libro, Precio_Libro. Esta estructura reduciría el número de uniones necesarias para las consultas y, por lo tanto, mejoraría el rendimiento de lectura.

Sin embargo, esta desnormalización también introduce redundancia y complica las operaciones de escritura. Por ejemplo, si el precio de un libro cambia, ahora necesitamos actualizar el Precio_Libro en todas las filas de la tabla "Ventas" que corresponden a ese libro. A pesar de esto, si las operaciones de lectura son mucho más frecuentes y críticas para el rendimiento que las operaciones de escritura, esta desnormalización podría ser beneficiosa.