Validación de requerimientos y modelado de bases de datos

Breve descripción:

El componente ofrece un enfoque integral para diseñar y gestionar datos de manera eficiente. Abarca la caracterización de la información, el modelado de estructuras relacionales y no relacionales, y la documentación detallada del modelo. Proporciona herramientas y técnicas para optimizar el almacenamiento, garantizar la integridad y facilitar la escalabilidad.

Diciembre 2024

Tabla de contenido

Introducción

La validación de requerimientos y el modelado de bases de datos han adquirido una relevancia crítica en el desarrollo de sistemas eficientes y escalables. A medida que las organizaciones manejan volúmenes crecientes de datos, los diseñadores y desarrolladores han tenido que perfeccionar sus métodos y herramientas para garantizar la precisión y optimización en la gestión de la información. En este marco, se presenta un componente integral que abarca desde la caracterización detallada de la información hasta el diseño y documentación de modelos relacionales y no relacionales.

Uno de los aspectos preponderantes de este proceso es la caracterización precisa de los datos. Esto incluye la identificación de tipos, características y restricciones que definen la calidad de la información, asegurando que se cumplan los estándares de integridad y consistencia. La comprensión de estos elementos permite diseñar bases de datos que no solo sean robustas, sino también adaptables a las necesidades cambiantes de los proyectos.

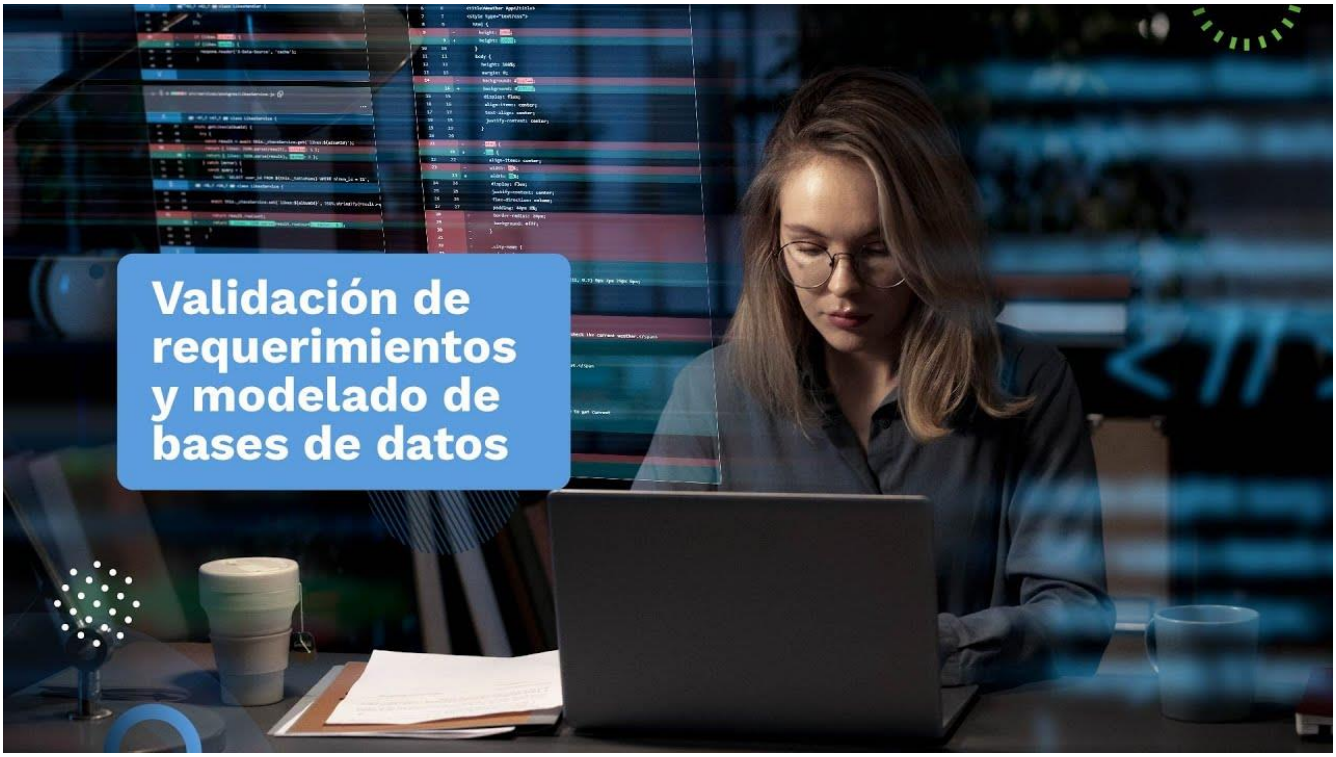
El siguiente enfoque fundamental es el modelado de la estructura de datos relacional, donde se detallan las técnicas para organizar tablas, definir relaciones y aplicar procesos de normalización. Este apartado prioriza la eficiencia y la minimización de redundancias, utilizando diagramas entidad-relación (ERD) y otros métodos visuales que facilitan la planificación y diseño de bases de datos bien estructuradas.

En paralelo, el modelado de estructuras de datos no relacionales aborda la flexibilidad y escalabilidad requeridas por aplicaciones modernas. Aquí se exploran bases de datos como documentos y grafos, con ejemplos prácticos de cómo manejar datos semiestructurados en formatos como JSON y XML. Este enfoque asegura que las soluciones sean ágiles y capaces de manejar grandes volúmenes de información distribuida.

Finalmente, el componente se enfoca en la documentación detallada del modelo de datos, proporcionando lineamientos claros para elaborar diccionarios de datos y diseñar esquemas bien definidos. La documentación no solo facilita el mantenimiento y las actualizaciones futuras, sino que también garantiza la colaboración efectiva entre equipos. Este recorrido por la validación y el modelado de datos te preparará para crear sistemas robustos y alineados con las mejores prácticas de la industria.

¡Bienvenido a este viaje por el mundo de la gestión de datos, donde aprenderá a diseñar estructuras eficientes y a documentar modelos de manera clara y precisa!

Video 1. Validación de requerimientos y modelado de bases de datos



Enlace de reproducción del video

Síntesis del video: Validación de requerimientos y modelado de bases de datos
En el componente formativo «Validación de requerimientos y modelado de bases de datos» se profundiza en los principios y técnicas para asegurar que los datos y requisitos de un sistema sean correctamente identificados, estructurados y documentados. Durante el desarrollo de este componente, se busca que el aprendiz adquiera un dominio sólido de la identificación y caracterización de datos, comprendiendo cómo definir sus tipos, restricciones y características.

Se abordan las mejores prácticas para la validación de requerimientos, con técnicas que aseguran que los requisitos sean claros, completos y cumplan con las expectativas del cliente o usuario.

El componente también enseña el modelado de la estructura de datos relacional, enfocándose en la creación de tablas, la definición de llaves primarias y foráneas, y la aplicación de procesos de normalización.

El modelado de la estructura de datos no relacional se aborda explicando las características y ventajas de bases de datos como las de documentos y grafos.

La documentación del modelo de datos es un paso crítico en la gestión de bases de datos, y el componente ofrece lineamientos detallados para elaborar diccionarios de datos y describir la estructura de manera clara.

Se exploran herramientas de modelado y ejemplos de cómo visualizar el diseño de una base de datos a través de diagramas entidad-relación (ERD), destacando su importancia para la planificación y comunicación de ideas complejas.

La validación de la estructura de datos también se discute, asegurando que el diseño final cumpla con los requerimientos y optimice el rendimiento del sistema.

A lo largo del componente, el aprendiz podrá asociar estos conceptos con aplicaciones prácticas de gestión de datos y documentación de requisitos, mejorando su capacidad para diseñar sistemas de manera precisa y efectiva.

¡Te invitamos a explorar los contenidos y desarrollar las habilidades necesarias para gestionar información de forma eficiente y confiable!

1. Caracterización de la información

La caracterización de la información es un paso fundamental en el diseño y modelado de bases de datos, ya que permite comprender cómo se deben organizar, procesar y almacenar los datos de manera eficiente y segura. Este proceso implica definir los conceptos de datos e información, identificar sus diferentes tipos y entender las características que determinan la calidad y utilidad de los datos. También se consideran las restricciones necesarias para mantener la integridad y seguridad de la base de datos.

1.1. Concepto de datos

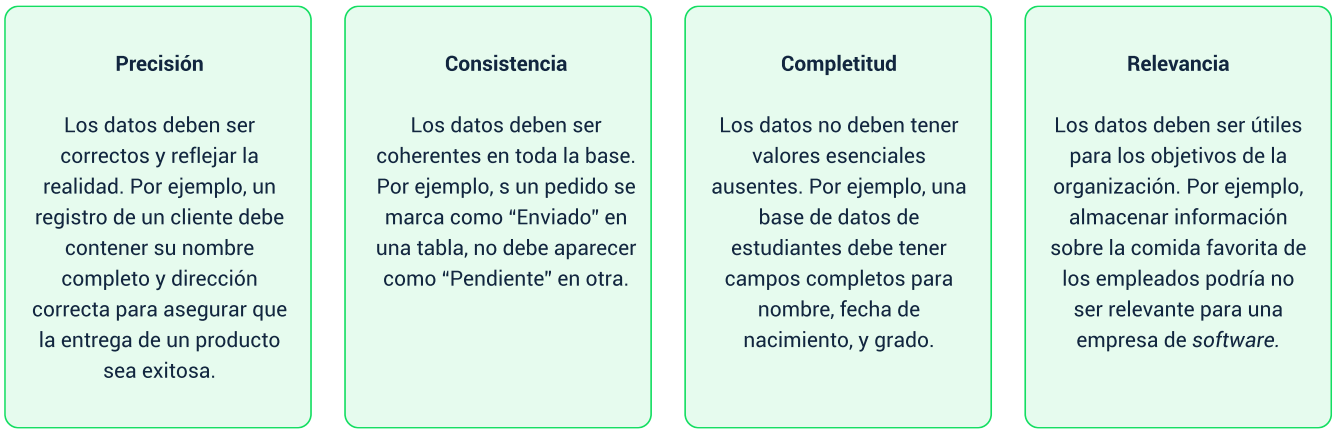
Los datos son unidades mínimas que representan hechos o cifras sin contexto, y su comprensión es importante para estructurar bases de datos adecuadamente.

- **Definición**: los datos pueden ser números (por ejemplo, 45), texto (como "Juan Pérez"), fechas (por ejemplo, 12/06/2023), o valores booleanos (verdadero o falso). Sin embargo, por sí solos, estos datos no proporcionan información útil hasta que se organizan y procesan.
- **Ejemplo práctico**: consideremos el dato "500". Sin un contexto, este número es solo una cifra, pero si se describe como "500 ventas realizadas en junio", se convierte en información significativa, útil para tomar decisiones de negocio.

1.2. Características de datos

La calidad de los datos depende de ciertas características que garantizan su utilidad y confiabilidad.

Figura 1. Características de los datos



Fuente. OIT, 2024.

1.3. Restricciones y clasificación

Las restricciones aseguran la integridad de los datos, mientras que la clasificación los organiza según criterios específicos.

a) **Restricciones**: estas son reglas que se aplican a los datos:

- **Restricción de unicidad**: garantiza que un valor no se repita. Por ejemplo, los números de seguridad social de los empleados deben ser únicos para evitar duplicados.
- **Restricción de tipo de datos**: define el tipo de datos que una columna puede contener. Una columna "Fecha de nacimiento" solo debe aceptar datos de tipo fecha.
- **Restricción de valor**: limita los valores posibles. Por ejemplo, una columna "Estado del pedido" podría aceptar solo "Pendiente", "Enviado" o "Entregado".

b) **Clasificación**: se organiza según criterios como sensibilidad o naturaleza:

- **Clasificación por sensibilidad**: los datos pueden ser públicos (como información general de contacto) o confidenciales (como datos financieros).
- **Clasificación por tipo de contenido**: pueden ser estáticos, como un archivo histórico, o dinámicos, como datos de transacciones en tiempo real.

1.4. Tipos de información

La información puede ser estructurada, no estructurada o semiestructurada, dependiendo de su organización y propósito.

- **Información estructurada**: esta se organiza en un formato predefinido, como tablas en una base de datos. Por ejemplo, una tabla de empleados podría tener columnas como ID, Nombre, Cargo, y Salario, lo que facilita su gestión y consulta mediante SQL.
- **Información no estructurada**: no sigue un patrón fijo y es difícil de organizar. Ejemplos incluyen correos electrónicos, documentos de texto o imágenes. Este tipo de información requiere técnicas avanzadas, como procesamiento de lenguaje natural (NLP), para su análisis.
- **Información semiestructurada**: tiene una estructura parcial o flexible, como archivos JSON o XML. Por ejemplo, un archivo JSON que almacena información de un cliente podría incluir campos como nombre, dirección, e historial de compras, pero la estructura puede variar entre registros.

2. Modelado de la estructura de datos relacional

El modelado de bases de datos relacionales es una práctica para organizar y estructurar datos de manera eficiente, garantizando que se puedan gestionar y consultar de forma óptima. Este proceso se basa en crear tablas con relaciones bien definidas, aplicando principios de normalización para reducir la redundancia y mantener la integridad de los datos. A lo largo de este tema, se explican en detalle los elementos para modelar datos relacionales, desde la selección del tipo de base de datos hasta los procesos técnicos de normalización.

2.1. Selección de bases de datos relacionales

Elegir la base de datos relacional adecuada es importante para un diseño eficiente y un rendimiento óptimo del sistema.

Factores a considerar:

La selección debe basarse en varios criterios, como la cantidad de datos que se espera manejar, el tipo de transacciones que se realizarán, y los requisitos de seguridad y escalabilidad. Por ejemplo, si se necesita un alto rendimiento en la consulta de grandes volúmenes de datos, una base de datos como PostgreSQL puede ser adecuada por su soporte robusto a operaciones complejas y su capacidad de manejar grandes cargas de trabajo.

Ejemplos de uso:

- **MySQL**: comúnmente utilizado en aplicaciones web que requieren una base de datos ligera y rápida. Por ejemplo, un sistema de blogs donde los posts y los usuarios están relacionados en tablas.

- **SQL Server**: preferido en entornos empresariales que demandan alta disponibilidad y seguridad, como un sistema de gestión de nómina de empleados.

Estos ejemplos destacan cómo la elección del DBMS debe alinearse con las necesidades específicas del proyecto, asegurando que la base de datos seleccionada pueda satisfacer los requisitos técnicos y operativos.

2.2. Análisis de relaciones y cardinalidades

El análisis de las relaciones y las cardinalidades entre tablas es fundamental para diseñar un esquema de datos efectivo.

Tipos de relaciones:

- **Uno a uno (1:1)**: una relación donde un registro en una tabla está vinculado a un único registro en otra tabla.
- **Uno a muchos (1)**: una relación donde un registro en una tabla está vinculado a múltiples registros en otra.
- **Muchos a muchos (M)**: una relación donde múltiples registros en una tabla pueden estar relacionados con múltiples registros en otra.

Figura 2. Tipos de relaciones

Fuente. OIT, 2024.

Tabla 1. Ejemplo de tabla de entrada y salida

Relación Uno a Uno (1:1)	Relación Uno a Muchos (1:N)	Relación Muchos a Muchos (N:M)
• Cada registro en la tabla A se relaciona con un único registro en la tabla B. • La relación es única en ambas direcciones. • Se usa para datos únicos y exclusivos.	• Un registro en la tabla A puede relacionarse con varios registros en la tabla B. • Cada registro en la tabla B pertenece a un único registro en la tabla A. • Es el tipo de relación más común.	• Varios registros en la tabla A pueden relacionarse con varios registros en la tabla B. • Requiere una tabla intermedia para establecer las relaciones. • La tabla intermedia contiene las claves primarias de ambas tablas.

Fuente: OIT, 2024.

Cardinalidad:

Describe cuántas veces una entidad puede estar asociada con otra.

- **Ejemplo práctico**: si en un sistema de ventas un cliente puede realizar varios pedidos, pero cada pedido pertenece a un solo cliente, la relación es

de uno a muchos (1). Sin embargo, si un pedido puede tener múltiples productos y cada producto puede aparecer en varios pedidos, la relación es muchos a muchos (M), lo que requeriría una tabla intermedia como "Detalles de pedidos".

2.3. Diseño de tablas y elementos

El diseño adecuado de tablas ayuda a organizar los datos y facilitar su uso en aplicaciones.

a) Elementos del diseño:

- **Campos**: los campos son las columnas de una tabla que definen el tipo de datos que se almacenan. Por ejemplo, en una tabla de "Clientes", los campos pueden incluir Nombre, Correo electrónico, Dirección, y Teléfono. Cada campo debe tener un tipo de dato específico para asegurar la consistencia y precisión.
- **Registros**: son las filas de la tabla que representan instancias específicas de datos. Por ejemplo, un registro en una tabla de "Productos" podría contener los datos de un producto específico, como ID del producto: 123, Nombre: Lápiz HB, y Precio: 0.50 USD.

b) Tipos de llaves:

- **Llave primaria**: identifica de manera única cada registro en una tabla. Ejemplo: en una tabla de "Empleados", la ID de empleado sería la llave primaria, garantizando que cada empleado tenga un identificador único.

- **Llave compuesta**: utiliza más de un campo para asegurar la unicidad. Ejemplo: en una tabla de "Reservas de hotel", una combinación de ID de habitación y Fecha de reserva puede formar una llave compuesta para evitar duplicados.
- **Llave foránea**: establece relaciones entre tablas, conectando campos de una tabla con la llave primaria de otra. Ejemplo: en una tabla de "Facturas", el ID de cliente actúa como una llave foránea que enlaza con la tabla de "Clientes".

Este diseño garantiza que las relaciones entre tablas se mantengan claras y estructuradas, facilitando consultas eficientes y minimizando errores de datos.

2.4. Procesos de normalización

La normalización es un conjunto de reglas y técnicas que organiza los datos en tablas para reducir la redundancia y garantizar la integridad de los datos.

- **Primera forma normal (1FN)**: los datos deben ser atómicos y no contener listas ni conjuntos en un solo campo. Ejemplo: una tabla de "Contactos" no debe tener un campo que liste varios números de teléfono separados por comas. En lugar de eso, cada número debe estar en un registro separado.
- **Segunda forma normal (2FN)**: se aplica a tablas que ya cumplen con 1FN, asegurando que cada campo no clave dependa completamente de la llave primaria. Ejemplo: en una tabla de "Pedidos", los detalles del cliente, como el nombre o la dirección, se trasladan a una tabla separada de "Clientes" para evitar duplicación.

- **Tercera forma normal (3FN)**: elimina las dependencias transitivas, asegurando que los datos estén organizados de forma óptima. Ejemplo: si una tabla de "Productos" incluye información sobre la categoría del producto y la descripción de la categoría, estos datos se mueven a una tabla de "Categorías" separada, con una relación entre ambas tablas.

Ejemplo práctico: imagine un sistema de gestión escolar donde los datos de los estudiantes y sus calificaciones se almacenan en una tabla grande. Aplicar procesos de normalización implica dividir esta tabla en varias, como "Estudiantes", "Cursos", y "Calificaciones", asegurando que cada tabla almacene solo la información relevante y esté bien relacionada con las demás.

3. Modelado de la estructura de datos no relacional

El modelado de datos no relacional ha ganado importancia con el auge de las aplicaciones modernas que requieren alta flexibilidad y escalabilidad. A diferencia de las bases de datos relacionales, las no relacionales almacenan datos sin una estructura rígida de tablas, lo que las hace ideales para manejar información semiestructurada o no estructurada, como documentos, archivos JSON o datos en memoria. Este tema explora los diferentes tipos de bases de datos no relacionales, sus estructuras, y las herramientas que permiten modelar y gestionar eficientemente estos datos.

3.1. Tipos de bases de datos no relacionales

Las bases de datos no relacionales (o NoSQL) se clasifican en varias categorías, cada una diseñada para casos de uso específicos.

- **Bases de datos de tipo clave-valor**: estas bases almacenan datos como pares clave-valor, donde cada clave es única y se asocia con un valor. Son útiles para almacenar datos simples y rápidos de recuperar, como sesiones de usuario. Ejemplo: Redis y Amazon DynamoDB son bases de datos clave-valor que se utilizan ampliamente para cachés y sistemas de búsqueda rápida.
- **Bases de datos de documentos**: almacenan datos en documentos, típicamente en formato JSON, BSON o XML. Son muy útiles para aplicaciones que manejan datos semiestructurados y dinámicos. Ejemplo: MongoDB permite almacenar y consultar documentos JSON completos, lo que es ideal para aplicaciones web que requieren una estructura de datos flexible.

- **Bases de datos de grafos**: estas bases se usan para representar y consultar relaciones complejas entre entidades. Son adecuadas para aplicaciones como redes sociales, donde las conexiones entre personas deben gestionarse eficientemente. Ejemplo: Neo4j es una base de datos de grafos que se utiliza para analizar redes sociales o sistemas de recomendación.
- **Bases de datos orientadas a columnas**: almacenan datos en columnas en lugar de filas, lo que mejora el rendimiento de consultas analíticas. Son útiles en aplicaciones de análisis de grandes volúmenes de datos, como en la analítica de big data. Ejemplo: Apache Cassandra y HBase son bases de datos orientadas a columnas que se utilizan en sistemas que requieren escalabilidad masiva y velocidad en la escritura y lectura de datos.

Ejemplo: un sistema de comercio electrónico que necesita almacenar información diversa sobre productos, carritos de compra y usuarios podría utilizar una base de datos de documentos como MongoDB. Los documentos pueden tener estructuras diferentes para productos y carritos, sin necesidad de una estructura fija como en las bases de datos relacionales.

3.2. Estructuras de datos en JSON y XML

Los formatos JSON y XML se utilizan ampliamente en bases de datos no relacionales para almacenar y transportar datos.

JSON (JavaScript Object Notation)

Es un formato ligero que representa datos como objetos clave-valor, lo que lo hace fácil de leer y escribir para humanos y máquinas. Se usa comúnmente en aplicaciones web y servicios RESTful.

Ejemplo: Un documento JSON que describe un producto puede contener campos como:

```
{
    "id": 101,
    "nombre":  "camiseta deportiva",
    "precio": "25.99
    "tallas": ["S", "M", "L", "XL"],
    "en_stock": true
}
```

Aquí, el formato JSON permite manejar listas y valores booleanos de manera sencilla.

XML (Extensible Markup Language)

Es un formato más estructurado que utiliza etiquetas para definir elementos y se emplea en aplicaciones que requieren validación y estructura más rígida.

Ejemplo: Un archivo XML para un pedido podría verse así:

```
<pedido>
    <id>12345</id>
    <cliente>Juan Pérez</cliente>
    <productos>
        <producto>
```

```
            <nombre>Portátil</nombre>
            <precio>750.00</precio>
        </producto>
        <producto>
            <nombre>ratón inalámbrico</nombre>
            <precio>25.50</precio>
        </producto>
    </producto>
</pedido>
```

XML es útil en sistemas para la interoperabilidad entre distintas plataformas.

Ejemplo: en una aplicación móvil que consume datos de un servidor, los datos pueden recibirse en formato JSON porque es más ligero y se procesa rápidamente en los navegadores y aplicaciones móviles.

3.3. Herramientas de modelado

Existen varias herramientas que facilitan el diseño y la gestión de bases de datos no relacionales, cada una adaptada a un tipo específico de datos.

- **MongoDB Compass**: una herramienta gráfica para explorar y analizar datos almacenados en MongoDB. Permite visualizar documentos, ejecutar consultas y optimizar índices.

- **Neo4j Browser**: utilizada para modelar y consultar datos en una base de datos de grafos. Ofrece una interfaz interactiva donde se pueden ejecutar consultas y visualizar relaciones complejas entre nodos.
- **Apache Cassandra**: utiliza nodos distribuidos y es excelente para manejar grandes cantidades de datos en tiempo real. Herramientas como DataStax Studio permiten modelar y analizar datos orientados a columnas.
- **RedisInsight**: un panel de control para visualizar y analizar datos en Redis. Ayuda a optimizar el rendimiento y gestionar datos clave-valor de manera eficiente.

Ejemplo: en una empresa que maneja millones de transacciones financieras diarias, Apache Cassandra podría ser la herramienta preferida debido a su capacidad de escalar horizontalmente y manejar datos en múltiples servidores.

3.4. Ventajas y desventajas

El uso de bases de datos no relacionales tiene pros y contras que deben evaluarse antes de la implementación.

a) Ventajas:

- **Flexibilidad**: las bases de datos no relacionales pueden manejar datos semiestructurados, lo que permite cambios rápidos en el esquema sin afectar la base de datos completa. Por ejemplo, agregar un nuevo campo a un documento en MongoDB es sencillo.
- **Escalabilidad horizontal**: son diseñadas para distribuir datos en varios servidores, lo que las hace ideales para aplicaciones que requieren alta disponibilidad y grandes volúmenes de datos. Ejemplo: Facebook usa bases

de datos de grafos para manejar conexiones entre usuarios a escala masiva.

- **Alto rendimiento**: las operaciones de lectura y escritura suelen ser muy rápidas, especialmente en bases de datos clave-valor como Redis.

b) Desventajas:

- **Falta de soporte para transacciones complejas**: a diferencia de las bases de datos relacionales, las no relacionales a menudo carecen de soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), lo que puede ser un problema en aplicaciones financieras.
- **Consistencia eventual**: en sistemas distribuidos, algunas bases de datos NoSQL garantizan la consistencia eventual en lugar de la consistencia inmediata, lo que puede ser problemático para aplicaciones que requieren datos siempre precisos.
- **Curva de aprendizaje**: aprender y optimizar bases de datos no relacionales puede ser complicado, especialmente para aquellos acostumbrados a los sistemas relacionales.

Ejemplo: en una aplicación de redes sociales, la base de datos de grafos como Neo4j es ideal para gestionar las conexiones entre usuarios y las recomendaciones de amigos. Sin embargo, para manejar transacciones financieras, se preferiría un sistema que garantice la consistencia inmediata, como un DBMS relacional.

Tabla 2. Bases de datos relacionales vs bases de datos no relacionales

Aspecto	Bases de Datos Relacionales	Bases de Datos No Relacionales
Modelo de datos.	Utilizan un modelo basado en tablas, donde los datos se organizan en filas y columnas.	Utilizan modelos de datos flexibles como documentos, gráficos, clave-valor o columnas.
Esquema.	Tienen un esquema predefinido y rígido, donde se deben definir las tablas y las relaciones antes de agregar datos.	Tienen un esquema flexible o sin esquema, permitiendo agregar datos sin estructuras fijas.
Lenguaje de consulta.	Utilizan SQL (Structured Query Language) para definir y manipular datos.	Utilizan diferentes lenguajes de consulta según el tipo, como JSON, CQL, o Gremlin.
Escalabilidad.	Escalabilidad vertical: aumentan su rendimiento añadiendo más potencia al servidor.	Escalabilidad horizontal: pueden manejar grandes volúmenes de datos distribuyéndolos en múltiples servidores.
Transacciones.	Soportan transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) para garantizar la integridad de los datos.	Soportan transacciones BASE (Básicamente Disponible, Estado Suave, Eventual Consistencia), priorizando la disponibilidad y la flexibilidad.
Ejemplos.	MySQL, PostgreSQL, Oracle, Microsoft SQL Server.	MongoDB, Cassandra, Redis, Neo4j, Couchbase.
Uso principal.	Adecuadas para aplicaciones con relaciones complejas entre datos y que requieren consistencia estricta.	Ideales para aplicaciones que necesitan manejar grandes volúmenes de datos, con requisitos de flexibilidad y alta disponibilidad.

Aspecto	Bases de Datos Relacionales	Bases de Datos No Relacionales
Integridad de los datos.	Alta integridad debido a la estructura rígida y las restricciones relacionales.	Menor integridad por la flexibilidad, pero con mayor capacidad para manejar datos no estructurados.
Relaciones entre datos.	Administran relaciones mediante claves primarias y foráneas, permitiendo un modelado relacional robusto.	Pueden manejar relaciones, pero no de la manera tradicional; algunas usan referencias o estructuras embebidas.
Rendimiento.	Rendimiento óptimo para operaciones transaccionales complejas y consultas estructuradas.	Mejor rendimiento para consultas rápidas y manejo de datos masivos con alta disponibilidad.

Fuente. OIT, 2024.

4. Documentación del modelo de datos

Documentar el modelo de datos es un paso en el desarrollo y mantenimiento de bases de datos. Una documentación clara y detallada asegura que los desarrolladores, analistas y otros miembros del equipo comprendan la estructura y el propósito de la base de datos, facilitando futuras actualizaciones, optimizaciones y la resolución de problemas. Este tema explora cómo diseñar y presentar la documentación de manera efectiva, abarcando desde el diseño hasta las recomendaciones para su implementación.

4.1. Diseño de la base de datos

El diseño de la base de datos abarca la estructura general y cómo se organizan los datos para garantizar su eficacia y rendimiento.

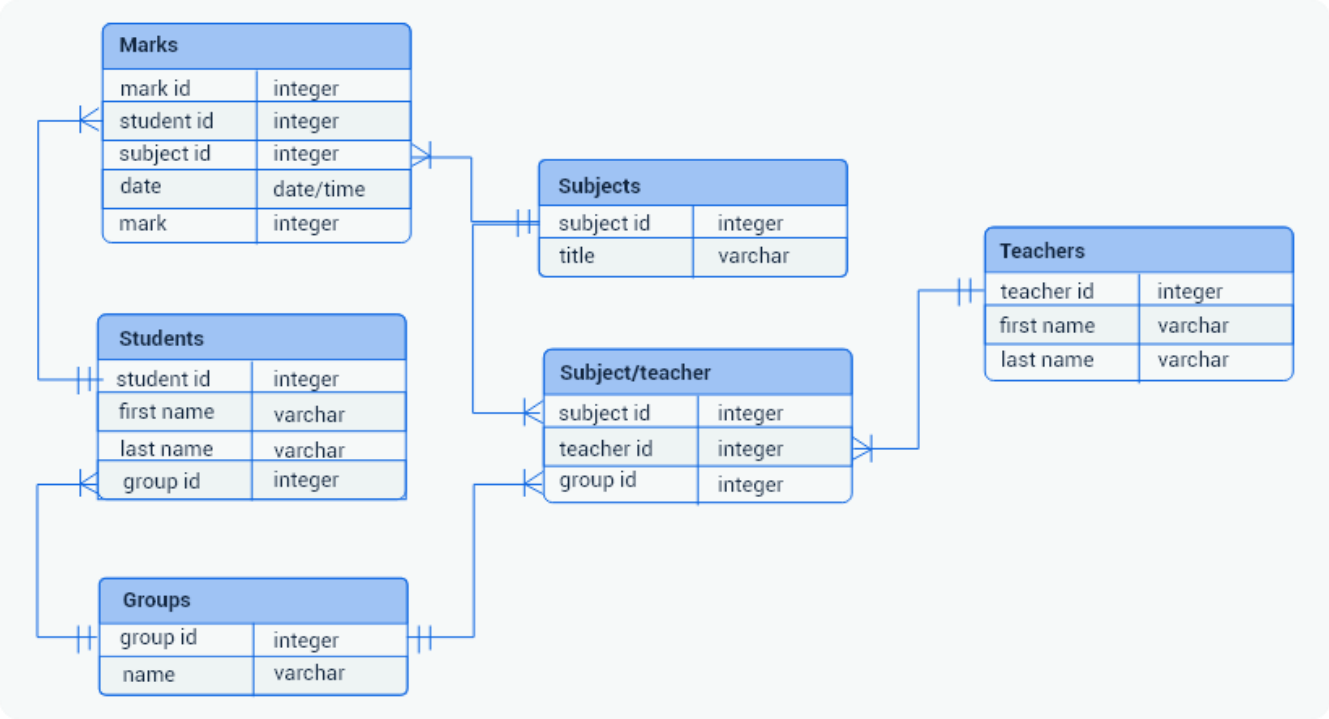
Esquema de la base de datos

Describe la estructura global, incluyendo tablas, columnas, relaciones y restricciones. El esquema debe mostrar cómo las diferentes entidades están interconectadas. Por ejemplo, en un sistema de gestión de inventario, el esquema podría incluir tablas como "Productos", "Proveedores", y "Pedidos", mostrando cómo estas tablas se relacionan mediante llaves foráneas.

Diagrama entidad-relación (ERD)

Es una representación gráfica del esquema de la base de datos. Los diagramas ERD ayudan a visualizar las entidades, los atributos y las relaciones entre ellas.

Figura 3. Ejemplo de un modelo entidad relación



Fuente. OIT, 2024.

Este modelo entidad-relación (ER) representa la estructura lógica y organizativa de una base de datos académica diseñada para gestionar y almacenar eficientemente información relacionada con distintos elementos del entorno educativo. En particular, el modelo organiza datos sobre estudiantes, grupos a los que pertenecen, asignaturas que estudian, calificaciones que obtienen, y los profesores que imparten las materias. Cada entidad en el modelo está interconectada mediante relaciones específicas que garantizan la integridad y coherencia de los datos, permitiendo que el sistema pueda realizar consultas y operaciones de manera eficiente.

A continuación, se explican los componentes clave del modelo:

Entidades y sus atributos

a) **Marks (Calificaciones)**

- **mark id (integer)**: identificador único para cada registro de calificación.
- **student id (integer)**: referencia al estudiante que recibió la calificación, se conecta con la entidad Students.
- **subject id (integer)**: referencia a la asignatura evaluada, se conecta con la entidad Subjects.
- **date (date/time)**: fecha en la que se registró la calificación.
- **mark (integer)**: la calificación obtenida por el estudiante en la asignatura específica.

b) **Students (Estudiantes)**

- **student id (integer)**: identificador único del estudiante.
- **first name (varchar)**: primer nombre del estudiante.
- **last name (varchar)**: apellido del estudiante.
- **group id (integer)**: referencia al grupo al que pertenece el estudiante, se conecta con la entidad Groups.

c) **Groups (Grupos)**

- **group id (integer)**: identificador único del grupo.
- **name (varchar)**: nombre o designación del grupo (por ejemplo, Grupo A, Grupo B).

d) **Subjects (Asignaturas)**

- **subject id (integer)**: identificador único de la asignatura.
- **title (varchar)**: nombre o título de la asignatura (por ejemplo, Matemáticas, Historia).

e) **Teachers (Profesores)**

- **teacher id (integer)**: identificador único del profesor.
- **first name (varchar)**: primer nombre del profesor.
- **last name (varchar)**: apellido del profesor.

f) **Subject/teacher (Asignatura/Profesor)**

- **subject id (integer)**: referencia a la asignatura impartida, se conecta con Subjects.
- **teacher id (integer)**: referencia al profesor que enseña la asignatura, se conecta con Teachers.
- **group id (integer)**: referencia al grupo al que se le imparte la asignatura, se conecta con Groups.

Relaciones

- **Relación entre Marks y Students**: la entidad Marks tiene un atributo student id que se conecta con student id en Students. Esto significa que cada calificación está asociada a un estudiante específico.
- **Relación entre Marks y Subjects**: la entidad Marks se conecta con Subjects a través del subject id. Cada calificación pertenece a una asignatura específica.
- **Relación entre Students y Groups**: la entidad Students tiene un atributo group id que se conecta con Groups, indicando el grupo al que pertenece el estudiante.
- **Relación entre Subject/teacher y Subjects, Teachers y Groups**: la entidad Subject/teacher actúa como una entidad de unión que relaciona

asignaturas con profesores y grupos. Esto permite indicar qué profesor imparte qué asignatura a cuál grupo.

- **Relación entre Teachers y Subject/teacher**: la entidad Teachers se conecta con Subject/teacher a través del teacher id, especificando qué profesor enseña una asignatura específica a un grupo.

Descripción general

Este modelo permite gestionar un sistema académico donde se pueden:

- Registrar calificaciones de estudiantes para distintas asignaturas.
- Asociar estudiantes a grupos específicos.
- Asignar profesores a materias y especificar los grupos a los que enseñan.
- Facilitar consultas sobre las calificaciones de estudiantes, asignaturas impartidas por profesores, y la distribución de estudiantes en grupos.

Definición de elementos

Cada tabla y sus atributos deben documentarse detalladamente. Esto incluye la descripción del propósito de la tabla y de cada columna, así como los tipos de datos y las restricciones aplicadas. Por ejemplo, una tabla de "Usuarios" podría tener un campo "Correo electrónico" con una restricción de unicidad para asegurar que no se repitan las direcciones.

4.2. Elaboración del diccionario de datos

El diccionario de datos proporciona una referencia completa de los elementos de la base de datos y sus atributos.

- **Descripción de tablas y columnas**: cada tabla debe tener una entrada en el diccionario que describa su función y la de cada columna. Por ejemplo, para una tabla "Clientes", se incluiría información como: Nombre: "Almacena el nombre completo del cliente"; Correo electrónico: "Dirección de correo única para contactar al cliente".
- **Tipos de datos y restricciones**: se especifican los tipos de datos (texto, numérico, fecha, etc.) y las restricciones aplicadas, como restricciones de unicidad, llaves primarias y llaves foráneas. Por ejemplo, un campo "Fecha de nacimiento" se documentaría con un tipo de dato "Fecha" y con una nota que indique que no se aceptan fechas futuras.
- **Índices y optimización**: si se han creado índices para mejorar el rendimiento de las consultas, estos también deben incluirse en el diccionario. Por ejemplo, un índice sobre el campo "ID de producto" en una tabla "Productos" para optimizar las búsquedas.

Ejemplo: en un sistema de ventas, un diccionario de datos ayudaría a los desarrolladores a comprender que el campo "Precio" en la tabla "Productos" es de tipo decimal y está restringido a valores positivos, evitando posibles errores durante las actualizaciones de la base de datos.

4.3. Estructura del modelo de datos

La estructura del modelo de datos define cómo se organizan y relacionan las entidades dentro de la base de datos.

- **Relaciones entre entidades**: se explican las conexiones lógicas entre tablas, como relaciones uno a uno, uno a muchos y muchos a muchos. Se deben

describir las llaves foráneas que gestionan estas relaciones. Por ejemplo, en un sistema de gestión de empleados, una tabla "Departamentos" estaría relacionada con "Empleados" mediante una llave foránea que conecta el "ID del departamento" en ambas tablas.

- **Integridad referencial**: se especifica cómo se mantienen las relaciones consistentes. Por ejemplo, se documenta que, si se elimina un registro en la tabla "Clientes", todos los registros relacionados en la tabla "Pedidos" deben actualizarse o eliminarse automáticamente según las reglas de la base de datos.
- **Manejo de datos complejos**: si se gestionan datos complejos, como listas anidadas o estructuras JSON, se debe explicar cómo se almacenan y acceden. Por ejemplo, un campo que almacene detalles del pedido en formato JSON debe tener instrucciones claras para la consulta y actualización de estos datos.

4.4. Recomendaciones de implementación

Las recomendaciones aseguran que la base de datos funcione de manera eficiente y se mantenga segura y fácil de actualizar.

- **Optimización de consultas**: se sugieren prácticas para mejorar el rendimiento, como el uso adecuado de índices, la desnormalización en casos específicos, o la partición de tablas cuando se manejan grandes volúmenes de datos. Por ejemplo, en una base de datos de transacciones financieras, se podría recomendar la creación de índices en campos de búsqueda frecuente como "Fecha de transacción".

- **Gestión de seguridad**: se incluyen pautas para proteger los datos, como el cifrado de datos sensibles, la gestión adecuada de permisos de usuario, y el uso de autenticación robusta. Por ejemplo, se recomienda que los campos como "Contraseña" se almacenen cifrados con algoritmos seguros como bcrypt.
- **Mantenimiento y respaldo**: se detallan estrategias para realizar copias de seguridad periódicas y planes de recuperación ante fallos. Por ejemplo, en un entorno crítico como un sistema de salud, se podría recomendar una estrategia de respaldo incremental diario y un respaldo completo semanal, con pruebas regulares de restauración.
- **Actualizaciones y escalabilidad**: se sugieren prácticas para facilitar actualizaciones futuras y garantizar que la base de datos pueda escalar con el crecimiento del sistema. Esto incluye el diseño modular y la previsión de posibles expansiones. Por ejemplo, si se espera que una base de datos crezca significativamente, se podría recomendar la implementación de particionamiento horizontal para distribuir datos en múltiples servidores.

Ejemplo: en un sistema de comercio electrónico que espera un aumento en el tráfico durante temporadas de venta, las recomendaciones podrían incluir el uso de un sistema de caché para mejorar la velocidad de las consultas y estrategias de balanceo de carga para gestionar múltiples solicitudes.

Síntesis

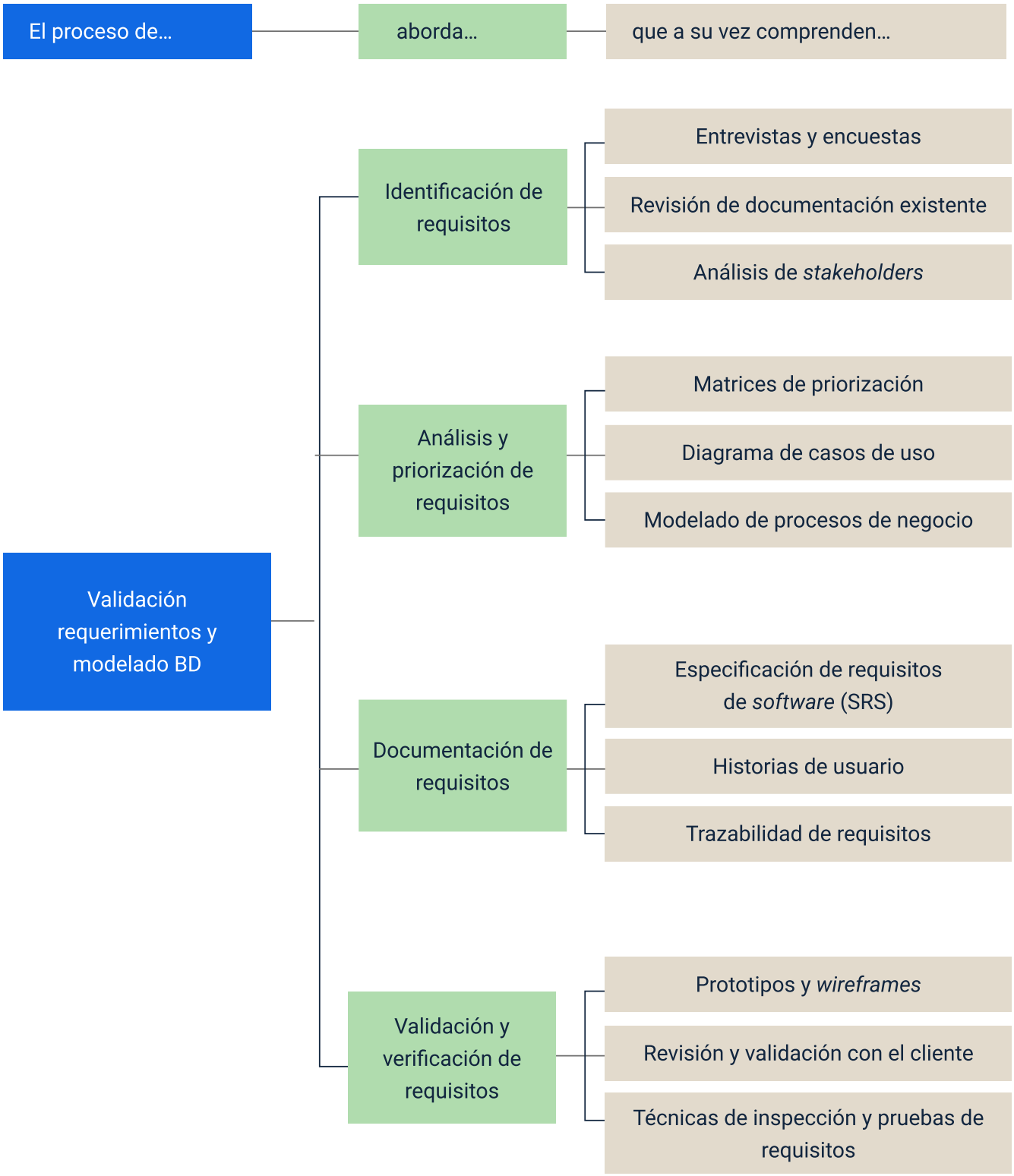
El siguiente diagrama ofrece una visión sintetizada de los principales conceptos y procedimientos desarrollados en el componente “Validación de requerimientos y modelado de bases de datos”. Está diseñado para facilitar al aprendiz la comprensión de la relación entre los diferentes elementos que componen este proceso integral de gestión de datos y requisitos.

En el centro del diagrama se encuentra el concepto principal relacionado con la validación de requerimientos y modelado de bases de datos, del cual se derivan las siguientes áreas: identificación y caracterización de datos, modelado de la estructura relacional, modelado de la estructura no relacional, y documentación del modelo de datos. Cada una de estas áreas se detalla en subtemas que ilustran el enfoque y contenido del componente, desde la recolección y validación de información hasta el diseño y documentación técnica.

El área de identificación y caracterización de datos abarca la definición de tipos de datos, restricciones, y características, subrayando la importancia de entender y estructurar adecuadamente la información. El modelado de la estructura relacional se enfoca en diseñar tablas, establecer relaciones claras mediante llaves primarias y foráneas, y aplicar procesos de normalización para optimizar el almacenamiento. Por otro lado, el modelado de la estructura no relacional aborda bases de datos como las de documentos y grafos, explicando cómo manejar datos semiestructurados y elegir la mejor tecnología para casos específicos. Finalmente, la documentación del modelo de datos incluye la creación de diccionarios de datos, la descripción detallada de

elementos, y recomendaciones para garantizar la transparencia y efectividad en la gestión de la base de datos.

Este diagrama sirve como una herramienta visual para navegar por los conceptos presentados en el componente, ayudando al aprendiz a entender rápidamente la secuencia y conexión de los procesos. Se invita a explorar el diagrama como un recurso complementario, ofreciendo una referencia rápida y un recordatorio estructurado de los elementos críticos en la validación y modelado de bases de datos.


El proceso de...
aborda...
que a su vez comprenden...
Validación requerimientos y modelado BD
Identificación de requisitos
Entrevistas y encuestas
Revisión de documentación existente
Análisis de *stakeholders*
Análisis y priorización de requisitos
Matrices de priorización
Diagrama de casos de uso
Modelado de procesos de negocio
Documentación de requisitos
Especificación de requisitos de *software* (SRS)
Historias de usuario
Trazabilidad de requisitos
Validación y verificación de requisitos
Prototipos y *wireframes*
Revisión y validación con el cliente
Técnicas de inspección y pruebas de requisitos

Material complementario

Tema	Referencia	Tipo de material	Enlace del recurso
1. Caracterización de la información	Ecosistema de Recursos Educativos Digitales SENA. (2023, marzo 26). *Fundamentos para entender las bases de datos*.	Video	https://www.youtube.com/watch?v=U8hJwMXfI1Q
1. Caracterización de la información	Ecosistema de Recursos Educativos Digitales SENA. (2021, mayo 23). *Introducción bases de datos*.	Video	https://www.youtube.com/watch?v=m-h80fyUIiA
2. Modelado de la estructura de datos relacional	Ecosistema de Recursos Educativos Digitales SENA. (2023, marzo 27). *Conceptos y estructuras de las bases de datos*.	Video	https://www.youtube.com/watch?v=vq14ny7O1WU
2. Modelado de la estructura de datos relacional	Ecosistema de Recursos Educativos Digitales SENA. (2022, octubre 11). *Conceptos y estructuras de las bases de datos*.	Video	https://www.youtube.com/watch?v=xUpr20u9dmc
2. Modelado de la estructura de datos relacional 3. Modelado de la estructura de datos no relacional	Ecosistema de Recursos Educativos Digitales SENA. (2021, diciembre 10). *Bases de datos relacionales y no relacionales*.	Video	https://www.youtube.com/watch?v=r97Ko4ZvIDQ

Tema	Referencia	Tipo de material	Enlace del recurso
3. Modelado de la estructura de datos no relacional	Ecosistema de Recursos Educativos Digitales SENA. (2022, diciembre 24). *Introducción, diseño y ejecución de pruebas BD*.	Video	https://www.youtube.com/watch?v=u83s6_Gj_q4
4. Documentación del modelo de datos	Ecosistema de Recursos Educativos Digitales SENA. (2022, septiembre 19). *Implementación y Gestión de Bases de datos*.	Video	https://www.youtube.com/watch?v=EXutOmlBaQQ

Glosario

Atributo: característica o propiedad de una entidad. Por ejemplo, un "Cliente" puede tener atributos como Nombre y Dirección.

Base de datos: conjunto organizado de datos almacenados y gestionados electrónicamente en un sistema, permitiendo acceso, manipulación y consulta eficiente.

Base de datos no relacional: sistema de gestión de bases de datos que almacena datos de forma no estructurada, como documentos, grafos o columnas, ideal para datos semiestructurados.

Base de datos relacional: sistema de gestión de bases de datos que organiza datos en tablas con relaciones bien definidas, basado en un modelo estructurado.

Consistencia: característica de los datos que asegura que la información sea coherente y uniforme en todas las tablas relacionadas de la base de datos.

Diagrama entidad-relación (ERD): gráfico que muestra las entidades, atributos y relaciones en un modelo de datos, ayudando a visualizar la estructura de la base de datos.

Diccionario de datos: repositorio detallado que describe las tablas, columnas, tipos de datos, restricciones y otros elementos de una base de datos.

Documentación: proceso de registrar y describir la estructura y elementos de la base de datos, facilitando la comprensión y mantenimiento del sistema.

Escalabilidad: capacidad de un sistema de base de datos para manejar un aumento en la carga de trabajo, ya sea mediante la adición de recursos (escalabilidad vertical u horizontal).

Esquema de la base de datos: representación estructural de cómo se organizan y relacionan las tablas y entidades en una base de datos.

Índice: estructura que mejora la velocidad de las consultas en una base de datos, permitiendo búsquedas más rápidas mediante el ordenamiento de los datos en un campo.

Integridad referencial: conjunto de reglas que garantizan que las relaciones entre tablas sean consistentes, evitando datos huérfanos o inconsistentes.

JSON: formato ligero para el intercambio de datos que representa objetos como pares clave-valor, muy utilizado en aplicaciones web.

Llave foránea: campo en una tabla que establece una relación con la llave primaria de otra tabla, manteniendo la integridad referencial.

Llave primaria: campo único que identifica de manera exclusiva cada registro en una tabla, asegurando que no existan duplicados.

Normalización: proceso de organización de datos en tablas para reducir la redundancia y mejorar la integridad de los datos mediante la aplicación de reglas específicas.

Primera forma normal (1FN): nivel básico de normalización que asegura que los datos sean atómicos y no contengan listas o conjuntos en un solo campo.

Restricción: regla aplicada a un campo en la base de datos para asegurar la validez y consistencia de los datos, como restricciones de unicidad o tipo de dato.

XML: lenguaje de marcado que define reglas para codificar documentos en un formato que tanto humanos como máquinas puedan leer.

Referencias bibliográficas

Coronel, C., & Morris, P. (2011). Diseño de bases de datos: conceptos y modelos. McGraw-Hill

Date, C. J. (2001). Introducción a los sistemas de bases de datos (7ª ed.). Addison-Wesley.

Efendy, Z. (2018). Normalization in database design. Jurnal CoreIT: Jurnal Hasil Penelitian Ilmu Komputer Dan Teknologi Informasi, 4(1), 34. https://doi.org/10.24014/coreit.v4i1.4382

Elmasri, R., & Navathe, S. B. (2010). Sistemas de bases de datos. Pearson Educación.

Fernandy, M., Darmawan, K. R., & Kristiyanto, D. Y. (2023). Comparison Analysis of Native Database Design with Object Oriented Design. Journal of Dinda: Data Science, Information Technology, and Data Analytics, 3(1), 6–10. https://doi.org/10.20895/dinda.v3i1.707

Hewasinghage, M., Nadal, S., Abelló, A., & Zimányi, E. (2023). Automated database design for document stores with multicriteria optimization. Knowledge and Information Systems, 65(7), 3045–3078. https://doi.org/10.1007/s10115-023-01828-3

Kunii, T. L., & Kunii, H. S. (1977). DATABASE DESIGN. In Proceedings of the Hawaii International Conference on System Sciences (pp. 200–203).

Letkowski, J. (2014). Doing database design with MySQL. Journal of Technology Research, 6(I), 1–15. Retrieved from

https://pdfs.semanticscholar.org/6dab/89a3e32c994cff67d0c0f40c69ef9446db2c.pdf%0A

Sadalage, P. J., & Fowler, M. (2013). NoSQL Distilled.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2006). Fundamentos de bases de datos (5ª ed.). McGraw-Hill.

Teorey, T. J., Lightstone, S. S., & Nadeau, T. P. (2006). Diseño de bases de datos: Modelado y diseño lógico (4ª ed.). Morgan Kaufmann.

Créditos

Elaborado por:

Organización Internacional del Trabajo